Efficient Cryptanalysis Techniques for Privacy-Preserving Record Linkage

Anushka Isuru Vidanage

A thesis submitted for the degree of
Doctor of Philosophy in Engineering and Computer Science
The Australian National University

Australian
National
University

December 2021

© Copyright by Anushka Isuru Vidanage 2021
All Rights Reserved

Except where otherwise indicated, this thesis is my own original work.

Anushka Isuru Vidanage
21 December 2021

To my Mum and Dad, who always put my needs over theirs, who raised me to be a better person, and who gave me the strength to reach for the stars. I hope that I have fulfilled their dreams.

To my beloved wife Dilini, without whom I could not have completed this work. Her continuous support, encouragement, and patience inspired me to achieve my goals.

Acknowledgements

"If I have seen further it is by standing on the shoulders of giants." – Isaac Newton (1642–1727)

Embarking my Ph.D has been a turning point in my life and achieving it would not have been possible without the support, inspiration, and guidance of many who offered their invaluable time and effort.

First and foremost, I would like to express my heartfelt gratitude to professor Peter Christen, my primary supervisor, for his selfless care, continuous guidance, and irreplaceable advice throughout my Ph.D, and for being the best supervisor that I could have ever asked for. Peter, you always inspired me with your vast knowledge, passion for research, and interesting ideas. Working with you has truly been a privilege and I will forever be grateful for the wisdom you have shared, the qualities you have instilled in me, and the opportunities you have offered, not only in academia but also in personal life. Thank you!

I extend my sincere thanks to Dr Thilina Ranbaduge, my associate supervisor, whom I have been extremely fortunate to work with. Thilina, your constant encouragement and prompt responses to all my queries have guided me through this journey. I was continually amazed by your extensive knowledge in subject matter and I am grateful for your willingness to assist me in every situation. Thank you for being a brilliant supervisor and a wonderful friend.

My warmest gratitude also goes to professor Rainer Schnell, my associate supervisor, who has invariably been greatly supportive in providing constructive feedback on my research. Thank you Rainer, for sharing your expertise and giving me timely advice to improve the quality of my research. In addition, I would also like to express my thanks to Sean Randall and Sirintra Vaiwsri whom I have co-authored with, for their valuable contributions.

I am sincerely honoured to have been selected as a recipient of the Postgraduate Research Scholarship at the Australian National University (ANU) which provided me the necessary financial solidity to complete this research. A special thanks goes to the School of Computing at ANU and the staff for enabling a positive working environment and extending continuous support during my Ph.D.

I am most fortunate to be working with all my fellow colleagues and members of the Data Mining and Matching group at ANU. Jeffrey Fisher, Yichen Hu, Sirintra Vaiwsri, Charini Nanayakkara, Nishadi Kirielle, and Asara Senaratne, I would like to thank you all for the stimulating discussions, productive feedback on my work, and for the fun and exciting time we had during the last four years.

I have had the opportunity to work with and learn from a number of wonderful people at ANU for the past few years. I am extremely thankful to professor Matthew Manning and Dr Gabriel Wong from the ANU College of Arts and Social Sciences for having me as a part of their project, which gave me the much needed form of escape from my research. I am also grateful to the ANU Researcher Development team: Victoria Firth-Smith, professor Inger Mewburn, and Dr Cally Guerin for their invaluable support and encouragement which helped me to keep things in perspective and elevate my confidence.

Pursuing a Ph.D would have been all the more difficult without the enormous support of Dr Shiromi Arunathilake and my undergraduate supervisor Dr Chamath Keppitiyagama from the University of Colombo School of Computing. Being the first mentors of my higher education life, both of you have guided me through hard times inspiring hope and instilling great qualities in me. Thank you for showing me the way and helping me grow. I am also most grateful for all my teachers and mentors who have contributed to shaping me into the person I am today.

I feel blessed to have so many amazing friends who stood by my side motivating me and pushing me to chase my ambitions. I will always cherish the fun times we had. Thank you all for your awesome friendship.

I must express my humble gratitude to all the people who help facilitating as well as continuing free education in Sri Lanka. It has provided me an equal opportunity and equitable access to quality education which eventually enabled me to pursue this Ph.D. Thank you all!

No words can ever express my profound gratitude towards my beloved parents who raised me with so much love and care. Mum and Dad, I am indebted to you for all the sacrifices you have made and for continuously believing in my potential. I am beyond lucky to have the most amazing sister who were with me in all my life's adventures. Thank you for being there for me though thick and thin, and for being so selflessly kind towards me.

Last but not least, my dearest wife Dilini, thank you for holding my hand along the way. You have always been my strongest support and my inspiration through this journey. Without you I would not have even pursued this path nor would I be able to accomplish it. Your unconditional love and understanding made this dream a reality. Thank you for being my best friend and soul-mate in life.

"Out of clutter, find simplicity. From discord, find harmony. In the middle of difficulty lies opportunity."

– Albert Einstein
(1879–1955)

Abstract

The linking of records across databases has seen an increasing interest over the last few decades in domains ranging from national census and healthcare to crime and fraud detection. This is due to the ability of record linkage (RL) to improve data quality and facilitate advanced data mining. Due to the absence of unique entity identifiers across the databases to be linked, RL is generally based on quasi-identifying (QID) attribute values of entities, such as their names, addresses, and dates of birth. However, the use of such personal identifying information often leads to individual, ethical, and legal concerns associated with privacy and confidentiality.

Privacy-preserving record linkage (PPRL) seeks to develop techniques that allow linkage of databases without compromising the privacy of the entities whose records are being linked. In general, PPRL techniques encode and/or encrypt QID values in sensitive databases in order to protect the privacy of the entities while allowing accurate linkage of records using encoded and/or encrypted values. However, certain PPRL techniques, such as popular Bloom filter encoding, have shown to be susceptible to privacy attacks, and a number of such attacks have been proposed on PPRL techniques over the years. While these attacks reveal different weaknesses in PPRL techniques, they also have limitations including the requirement of knowledge by an adversary about specific parameters, and significant memory and time consumption. Therefore, further research into both analysing existing privacy attacks and exploring novel attack methods is vital to better understand the risks associated with real-world PPRL projects.

In this thesis we present a comprehensive research study about privacy attacks on PPRL. We start by proposing a taxonomy of attacks on PPRL in which the existing attacks are categorised under twelve dimensions. Our taxonomy can be used to analyse the characteristics of privacy attacks and identify their limitations. Next, we propose a framework to quantify the vulnerabilities associated with both plaintext and encoded values of a sensitive database. Such a framework will help data custodians to assess the privacy guarantees of various PPRL techniques applied on their own databases, and help them to make informed decisions based on that assessment.

We then propose three novel privacy attacks on different PPRL techniques which overcome the drawbacks of existing attacks. Our attacks exploit weaknesses of PPRL techniques in order to reidentify encoded QID values. Compared to previous attacks, our attacks require less knowledge of the encoding process, where one of the attacks can be applied on a diverse range of PPRL techniques. Finally, we conduct an experimental evaluation of all three of the proposed attack methods using real-world databases and compare them against several existing privacy attacks on PPRL. The results of this evaluation show that our attacks outperform existing attacks on PPRL in terms of reidentification accuracy and scalability to large databases.

List of Publications

- **Publications contributing to this thesis:**

  1. **A Graph Matching Attack on Privacy-Preserving Record Linkage:** *Anushka Vidanage*, Thilina Ranbaduge, Peter Christen, Rainer Schnell. In proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM), October 2020, Ireland, pp. 1485–1494. (Full paper) – corresponds to Chapter 8 of this thesis.
  2. **A Privacy Attack on Multiple Dynamic Match-key based Privacy-Preserving Record Linkage:** *Anushka Vidanage*, Thilina Ranbaduge, Peter Christen, Sean Randall. In International Journal of Population Data Science (IJPDS), August 2020, Vol. 5 No. 1. (Full paper) – corresponds to Chapter 6 of this thesis.
  3. **Efficient Pattern Mining based Cryptanalysis for Privacy-Preserving Record Linkage:** *Anushka Vidanage*, Thilina Ranbaduge, Peter Christen, Rainer Schnell. In proceedings of the 35th IEEE International Conference on Data Engineering, April 2019, Macau, pp.1698-1701. (Short paper) – corresponds to Chapter 7 of this thesis.
  4. **Pattern Mining based Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage:** Peter Christen, *Anushka Vidanage*, Thilina Ranbaduge, Rainer Schnell. In proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), June 2018, Melbourne, pp. 530-542. (Full paper) – corresponds to Chapter 7 of this thesis.
  5. **A Taxonomy of Attacks on Privacy-Preserving Record Linkage:** *Anushka Vidanage*, Thilina Ranbaduge, Peter Christen, Rainer Schnell. Under review, (Full paper) – corresponds to Chapter 3 of this thesis.

- **Minor contributions:**

  1. **A critique and attack on "Blockchain-based privacy-preserving record linkage"**: Peter Christen, Rainer Schnell, Thilina Ranbaduge, *Anushka Vidanage*. In Information Systems, 2021 pp. 101930.
  2. **Bloom Filter based Encoding Methods**: Peter Christen, Thilina Ranbaduge, Rainer Schnell, *Anushka Vidanage*, Christian Borgs (co-authored book chapter). In Linking Sensitive Data - Methods and Techniques for Practical Privacy-Preserving Information Sharing, Springer, 2020, ch. 8, pp. 193-219.

3. **Attacking and Hardening Bloom Filter Encoding**: Peter Christen, Thilina Ranbaduge, Rainer Schnell, *Anushka Vidanage*, Sirintra Vaiwsri (co-authored book chapter). In Linking Sensitive Data - Methods and Techniques for Practical Privacy-Preserving Information Sharing, Springer, 20-20, ch. 9, pp. 221-251.
4. **Empirical Evaluation**: Peter Christen, Thilina Ranbaduge, Rainer Schnell, *Anushka Vidanage*, Sirintra Vaiwsri (co-authored book chapter). In Linking Sensitive Data - Methods and Techniques for Practical Privacy-Preserving Information Sharing, Springer, 2020, ch. 12, pp. 323-344.
5. **Evaluating hardening techniques against cryptanalysis attacks on Bloom filter encodings for record linkage:** Thilina Ranbaduge, *Anushka Vidanage*, Sirintra Vaiwsri, Rainer Schnell, Peter Christen. In International Journal of Population Data Science (IJPDS), August 2018, Vol. 3 No. 4.

- **Other achievements:**

1. **Finalist:** Visualise Your Thesis (VYT) competition (university level), 2020. The Australian National University. – presentation available at: https://youtu.be/p2hoCgAQvNU
2. **People's choice award:** Three Minute Thesis (3MT) competition (university level), 2019. The Australian National University. – presentation available at: https://youtu.be/U8tn0ilYBJI
3. **Winner:** Three Minute Thesis (3MT) competition (college level), 2019. College of Engineering and Computer Science, The Australian National University.

Contents

Acknowledgements **vii**

Abstract **xi**

List of Publications **xiii**

List of Figures **xxii**

List of Tables **xxv**

List of Notation **xxvii**

List of Abbreviations **xxix**

1 Introduction **1**
1.1 Attacks on Privacy-Preserving Record Linkage 1
1.1.1 Importance of Information Privacy 2
1.1.2 Data Privacy Policies and Laws 3
1.2 Motivation of the Research 4
1.3 Aims of the Research 5
1.4 Contributions of the Research 7
1.5 Scope of the Research 8
1.6 Research Methodology 10
1.7 Thesis Outline 11

2 Background **13**
2.1 Record Linkage (RL) 13
2.2 Privacy-Preserving Record Linkage (PPRL) 14
2.3 The PPRL Process 15
2.4 Privacy Aspects of PPRL 18
2.4.1 The Roles of Parties 18
2.4.2 Number of Parties 19
2.4.3 Adversary Models 20
2.5 Encoding and Encryption Techniques in PPRL 22
2.5.1 Bloom Filter (BF) Encoding 22
2.5.2 Hashing based Techniques 24
2.5.3 Reference Value based Techniques 26
2.5.4 Embedding based Techniques 27

2.5.5 Differential Privacy based Techniques . . . 28
2.5.6 Numerical Encoding Techniques . . . 29
2.5.7 Secure Multi-party Computation based Techniques . . . 30
2.6 PPRL Evaluation Metrics . . . 31
2.6.1 Linkage Quality Metrics . . . 31
2.6.2 Scalability Metrics . . . 33
2.6.3 Privacy Metrics . . . 33
2.7 Summary . . . 35

3 A Taxonomy of Attacks on Privacy-Preserving Record Linkage 37
3.1 Introduction . . . 37
3.2 Contribution . . . 38
3.3 An Outline of Attacks on PPRL . . . 38
3.3.1 Underlying Assumptions . . . 39
3.3.2 The Main Steps of an Attack on PPRL . . . 40
3.4 Taxonomy Overview of Attacks on PPRL . . . 41
3.5 Adversarial Aspects . . . 42
3.5.1 Different Types of Adversaries . . . 43
3.5.2 Knowledge About Encoding Techniques and Parameters . . . 44
3.5.3 Availability and Accessibility to Data . . . 45
3.5.4 Non-Technical Adversary Issues . . . 49
3.6 Technical Aspects . . . 51
3.6.1 Attack Types . . . 51
3.6.2 Attack Scope . . . 53
3.6.3 Vulnerability . . . 54
3.6.4 Complexity . . . 54
3.7 Practical Aspects . . . 55
3.7.1 Success of an Attack . . . 55
3.7.2 Scalability of an Attack . . . 60
3.7.3 Implementation of an Attack . . . 61
3.7.4 Databases Used for Evaluation . . . 61
3.8 Summary . . . 63

4 Existing Attacks on Privacy-Preserving Record Linkage 65
4.1 Introduction . . . 65
4.2 Existing Privacy Attacks on PPRL . . . 66
4.2.1 Constraint Satisfaction based Cryptanalysis for Bloom Filters . . . 66
4.2.2 Frequency based Manual Cryptanalysis for Bloom Filters . . . 69
4.2.3 Optimisation based Cryptanalysis for Bloom Filters . . . 71
4.2.4 Frequency based Cryptanalysis for Bloom Filters . . . 72
4.2.5 Extended Frequency based Cryptanalysis for Bloom Filters . . . 74
4.2.6 Graph Traversal based Cryptanalysis for Bloom Filters . . . 76
4.2.7 Graph Matching based Attack for Similarity Tables . . . 78
4.3 Other Related Privacy Attacks . . . 79

4.4 Summary . . . 80

5 A Vulnerability Assessment Framework for Privacy-Preserving Record Linkage 81
5.1 Introduction . . . 81
5.2 Contribution . . . 82
5.3 Worst-case Attack Scenario . . . 83
5.4 Overview of the Vulnerability Assessment Framework . . . 84
5.5 Assessing Plaintext Vulnerabilities . . . 88
5.5.1 Vulnerability of a Single Value . . . 88
5.5.2 Vulnerability of a Pair of Values . . . 90
5.6 Assessing Encoded Vulnerabilities . . . 91
5.6.1 Vulnerability and Assignability of a Single Encoding . . . 92
5.6.2 Vulnerability and Assignability of a Pair of Encodings . . . 93
5.7 Relevance of Plaintext and Encoded Vulnerabilities . . . 94
5.8 Analysis of Existing Attacks on PPRL . . . 95
5.9 Experimental Evaluation . . . 97
5.9.1 Databases and Evaluation Setup . . . 97
5.9.2 Results and Discussion . . . 99
5.9.3 Recommendations . . . 102
5.10 Summary . . . 103

6 A Frequency based Privacy Attack on Multiple Match-Key Encoding 105
6.1 Introduction . . . 105
6.1.1 Multiple Match-Key Encoding . . . 105
6.1.2 Contribution . . . 108
6.2 Frequency based Privacy Attack on MMK Encoding . . . 108
6.2.1 Calculating Frequency Distributions . . . 109
6.2.2 Measuring the Correlation of Frequency Distributions . . . 110
6.2.3 Filtering Candidate Plaintext Match-keys . . . 112
6.2.4 Reidentifying Encoded Match-Keys . . . 113
6.2.5 Reidentification of Plaintext Values . . . 114
6.3 Complexity Analysis . . . 115
6.4 Experimental Evaluation . . . 116
6.4.1 Databases and MMK Encoding . . . 116
6.4.2 Evaluation Criteria . . . 117
6.4.3 Results and Discussion . . . 119
6.4.4 Privacy Improvements and Recommendations . . . 123
6.5 Practical Considerations of the MMK Encoding . . . 126
6.6 Summary . . . 126

7 Pattern-mining based Cryptanalysis on Bloom Filter Encoding 129
7.1 Introduction . . . 129
7.2 Contribution . . . 130
7.3 Overview of the Bit Pattern based Cryptanalysis Attack . . . 132
7.3.1 Preliminaries and Assumptions . . . 132
7.3.2 Probability of Collision When Hashing One Element into a Bloom Filter . . . 133
7.3.3 Applicability of Pattern-mining Techniques . . . 134
7.4 Pattern-mining based Frequent Q-gram Identification . . . 136
7.4.1 Identifying Co-occurring Bit Positions . . . 136
7.4.2 Finding Q-grams with Inaccurate Bit Positions . . . 138
7.5 Language Model based Q-gram Set Expansion . . . 139
7.5.1 Generating the Language Model Graph . . . 140
7.5.2 Identifying Further Q-grams using the Language Model . . . 141
7.5.3 Probability of False Positives for Bloom Filter Subsets . . . 143
7.5.4 Probability of Collisions between Non-Frequent and Frequent Bit Positions . . . 145
7.6 Partitioning based Q-gram Set Expansion . . . 146
7.6.1 Plaintext and Bloom Filter Partitioning . . . 146
7.6.2 Identifying Further Q-grams using the Partitions . . . 149
7.7 Reidentification of Plaintext Values . . . 150
7.7.1 Length Filtering Process . . . 151
7.8 Complexity Analysis . . . 152
7.9 Experimental Evaluation . . . 153
7.9.1 Databases and Experimental Setup . . . 153
7.9.2 Evaluation Criteria . . . 153
7.9.3 Results and Discussion . . . 154
7.9.4 Recommendations . . . 160
7.10 Summary . . . 161

8 A Graph Matching Attack on Privacy-Preserving Record Linkage 163
8.1 Introduction . . . 163
8.2 Contribution . . . 165
8.3 A Graph Matching based Privacy Attack . . . 166
8.3.1 Preliminaries and Assumptions . . . 167
8.3.2 Generating Similarity Graphs . . . 167
8.3.3 Node Feature Generation . . . 169
8.3.4 Node Matching . . . 170
8.3.5 Reidentification of Plaintext Values . . . 175
8.4 Adjusting Similarities . . . 175
8.5 Complexity Analysis . . . 177
8.6 Experimental Evaluation . . . 177
8.6.1 Databases and Experimental Setup . . . 177
8.6.2 Evaluation Criteria . . . 178

8.6.3 Results and Discussion . . . 180
8.6.4 Recommendations . . . 183
8.7 Summary . . . 184

9 A Comparative Evaluation of Attacks on Privacy-Preserving Record Linkage 185
9.1 Introduction . . . 185
9.2 Baseline Attack Methods . . . 186
9.3 Databases and Parameter Settings . . . 187
9.4 Evaluation Criteria . . . 191
9.5 Experimental Results . . . 192
9.6 Discussion . . . 199
9.7 Recommendations for PPRL . . . 200
9.7.1 Technical Recommendations . . . 200
9.7.2 Non-Technical Recommendations . . . 202
9.8 Summary . . . 203

10 Conclusion and Future Work 205
10.1 Outline of the Research Challenges . . . 205
10.2 Summary of the Contributions . . . 206
10.3 Limitations of the Research . . . 208
10.4 Future Research Directions . . . 209
10.5 Conclusion . . . 211

A Bloom Filter Encoding for Privacy-Preserving Record Linkage 213
A.1 Bloom Filter (BF) Encoding . . . 213
A.1.1 Bloom Filter Encoding Process . . . 214
A.1.2 Probability of False Positives . . . 214
A.1.3 Optimal Parameter Setting . . . 215
A.2 Bloom Filter Hashing Techniques . . . 216
A.2.1 Double Hashing (DH) . . . 216
A.2.2 Random Hashing (RH) . . . 217
A.3 Bloom Filter Encoding Techniques . . . 217
A.3.1 Attribute-Level Bloom Filters (ABF) . . . 217
A.3.2 Cryptographic Long-term Key (CLK) . . . 219
A.3.3 Record-Level Bloom Filters (RBF) . . . 219
A.4 Bloom Filter Hardening Techniques . . . 222
A.4.1 Salting . . . 222
A.4.2 Balancing . . . 222
A.4.3 XOR-Folding . . . 223
A.4.4 Window based XORing . . . 224
A.4.5 Re-sampling based XORing . . . 224

B Assessing Disclosure Risk in Microdata **225**
B.1 Types of Disclosure Risks 225
B.2 Assessing Disclosure Risk 225
B.3 Disclosure Control Methods 227

References **229**

List of Figures

1.1 The proposed research methodology. 9

2.1 Main steps of the PPRL process. 16
2.2 An example Bloom filter encoding process. 23

3.1 Overview of a privacy attack on PPRL. 39
3.2 A taxonomy of attacks on PPRL. 42
3.3 25 combinations of column-wise and row-wise overlap between the sensitive and plaintext databases. 46

4.1 Outline of the frequency attack proposed by Christen et al. in 2017. . . 73

5.1 Overall idea behind an encoding and attacking process in PPRL. 86

6.1 Multiple match-key encoding process. 107
6.2 First two steps of the privacy attack on multiple match-key encoding. . 109
6.3 Accuracy results of the attack on multiple match-key encoding. 118

7.1 Outline of the pattern-mining process in the attack on Bloom filter encoding. 131
7.2 Overview of the pattern-mining attack on Bloom filter encoding. 132
7.3 Example q-gram graph generation process for the second step of the pattern-mining attack. 140
7.4 Precision and recall results of identified q-grams in the pattern-mining attack. 155

8.1 Overview of the graph matching based attack on PPRL. 165
8.2 An example bi-partite graph matching problem. 172
8.3 An example solution of the bi-partite graph matching problem. 175
8.4 The relationship between encoded and plaintext similarities. 176
8.5 Accuracy results for graph matching attack without and with similarity adjustments applied. 179

9.1 Frequency distributions of different quasi-identifying attribute value combinations. 189
9.2 Frequency distributions of q-grams in quasi-identifying attribute value combinations. 190

A.1 An example illustration of the cryptographic long-term key encoding. . 218
A.2 An example illustration of record-level Bloom filter encoding. 220
A.3 An example of Bloom filter balancing process. 223

List of Tables

3.1 Specific notation used in Chapter 3 . . . 38
3.2 Different output settings of 1-to-1, 1-to-*many*, *many*-to-1, and *many*-to-*many* assignments. . . . 57
3.3 A characterisation of existing privacy attacks using the proposed twelve dimensions. . . . 62

4.1 Specific notation used in Chapter 4. . . . 66

5.1 Specific notation used in Chapter 5. . . . 83
5.2 An example encoding of a record using QID values, tokens, and token substrings of that record and using four PPRL encoding techniques. . . . 85
5.3 The application of vulnerability assessments for different plaintext values and encoding techniques. . . . 95
5.4 Vulnerabilities and the corresponding assignabilities exploited by various attacks on PPRL. . . . 96
5.5 The vulnerability results of the plaintext QID values, tokens, and token substrings. . . . 98
5.6 The co-occurrence vulnerability results of the plaintext QID values, tokens, and token substrings. . . . 99
5.7 The vulnerability results of the encoded values for Bloom filter, tabulation min-hash, and two-step hash encoding techniques. . . . 100
5.8 The assignability results of the encoded values for Bloom filter, tabulation min-hash, and two-step hash encoding techniques. . . . 101
5.9 The vulnerability / assignability results of the encoded values for statistical linkage key and multiple match-key encoding techniques. . . . 102

6.1 Specific notation used in Chapter 6. . . . 106
6.2 An example correlation matrix for an encoded match-key. . . . 113
6.3 An example alignment of match-key values with plaintext values. . . . 114
6.4 Attribute reidentification accuracy results for the frequency based attack. 120
6.5 Identity reidentification accuracy results for the frequency based attack. 121
6.6 Probability of suspicion results for the frequency based attack. . . . 122
6.7 Time and memory consumption analysis of different steps of the frequency based attack. . . . 123
6.8 Comparison of precision and recall of the linkage when proposed improvement is applied. . . . 124

7.1 Specific notation used in Chapter 7. . . . 130
7.2 Probabilities of obtaining a certain Hamming weight, when hashing a single q-gram multiple times. . . . 134
7.3 Probabilities of hashing a q-gram into the same 1-bit positions that were previously hit by other q-grams. . . . 145
7.4 Upper bound probabilities of other q-grams sharing 1-bit positions set previously by frequent q-grams. . . . 146
7.5 Summary results of the proposed attack for different attribute combinations. . . . 154
7.6 Attribute reidentification accuracy results for the pattern-mining attack. 156
7.7 Identity reidentification accuracy results for the pattern-mining attack. . 157
7.8 Probability of suspicion results for the pattern-mining attack. . . . 158
7.9 Summary results of the proposed refinement steps of the attack. . . . 159
7.10 Time and memory consumption analysis of different steps of the pattern-mining attack. . . . 160

8.1 Specific notation used in Chapter 8 166
8.2 The node, edge, and neighbourhood features generated for each node in graphs. . . . 169
8.3 Percentages of accurate attribute reidentifications using different bipartite graph matching and edge weighting methods. . . . 180
8.4 Attribute reidentification accuracy results for the graph matching attack. 181
8.5 Identity reidentification accuracy results for the graph matching attack. 182
8.6 Probability of suspicion results for the graph matching attack. . . . 183
8.7 Time and memory consumption analysis of different steps of the graph matching attack. . . . 184

9.1 Percentages of matching records and changes in QID attribute values in all databases pairs. . . . 188
9.2 Attribute reidentification accuracy results for all seven attack methods with the EURO database pairs. . . . 191
9.3 Identity reidentification accuracy results for all seven attack methods with the EURO database pairs. . . . 192
9.4 Attribute reidentification accuracy results for six attack methods with the NCVR and MVR database pairs. . . . 193
9.5 Identity reidentification accuracy results for six attack methods with the NCVR and MVR database pairs. . . . 193
9.6 Attribute reidentification accuracy results for three attack methods with the large NCVR database pairs. . . . 194
9.7 Identity reidentification accuracy results for three attack methods with the large NCVR database pairs. . . . 194
9.8 Probability of suspicion results for all seven attack methods with the EURO database pairs. . . . 195

9.9 Probability of suspicion results for all seven attack methods with the NCVR and MVR database pairs. 196
9.10 Attribute reidentification accuracy results for attack methods with the EURO database pairs when two hardening methods are applied. 197
9.11 Identity reidentification accuracy results for attack methods with the EURO database pairs when two hardening methods are applied. 197
9.12 Time and memory consumption analysis of different attack methods using different database pairs. 198

List of Notation

$\mathbf{G}$	Global population
$\mathbf{D}^s$	Sensitive database
$\mathbf{D}^p$	Plaintext database
$\mathbf{Q}^s$, $\mathbf{Q}^p$	Plaintext quasi-identifying (QID) attribute values in $\mathbf{D}^s$ and $\mathbf{D}^p$
$\mathbf{D}^e$	Encoded database
$\mathbf{E}$	Encoded QID attribute values in $\mathbf{D}^e$
$\mathbf{M}$	Microdata in the sensitive database $\mathbf{D}^s$
$\mathcal{E}^s$, $\mathcal{E}^p$	Real-world entities represented by the records in $\mathbf{D}^s$ and $\mathbf{D}^p$
$\mathbf{L}$	Linkage results (list of record identifier pairs)
$G = (V, E)$	A graph with a set of nodes V and a set of edges E
$\mathbf{s}$	Set of elements
$\mathbf{f}$	Frequency distribution
$\mathbf{b}$	Bloom filter
r	Record in a database
n	Number of record in a database
ρ	Probability
$l = \lvert\mathbf{b}\rvert$	Length of a Bloom filter
p	Bit position in a Bloom filter, with $1 \leq p \leq l$
q	Q-gram
l_q	Length of a q-gram
$encode()$	Encoding function
$attack()$	Attack function
$h()$	Hash function

List of Abbreviations

BF	Bloom filter
CLK	Cryptographic long-term key
DC	Data consumer
DH	Double hashing
DO	Database owner
HBC	Honest-but-curious
HMAC	Keyed-hash message authentication code
IA	Inside adversary
LU	Linkage unit
MA	Multi-actor adversary
MMK	Multiple match-key
MVR	Michigan voter registration
NCVR	North Carolina voter registration
OA	Outside adversary
PPRL	Privacy-preserving record linkage
PS	Probability of suspicion
QID	Quasi-identifier
RH	Random hashing
RL	Record linkage
SLK	Statistical linkage key
SMC	Secure multi-party computation
TMH	Tabulation min-hash
2SH	Two-step hash

Chapter 1

Introduction

In this chapter we introduce the work presented in this thesis by highlighting important aspects of our research. We first provide an introduction to the research topic in Section 1.1, where we also outline the importance of information privacy and existing laws and regulations put forth in different countries and by organisations with regard to protecting the privacy of sensitive information. Then we discuss the motivation of our research study in Section 1.2, and the aim of our research, along with research objectives, in Section 1.3. We next describe the contributions, scope, and the methodology of our research in Sections 1.4 to 1.6, respectively. Finally, we outline the structure of this thesis in Section 1.7.

1.1 Attacks on Privacy-Preserving Record Linkage

With increasing amounts of data being collected every day, the development of efficient and effective data integration techniques has recently gained popularity in diverse disciplines including academia, governments, and private industries [38]. Application domains including business intelligence, healthcare, national census, population informatics, fraud detection, and national security, can gain substantial benefits from the knowledge that integrated data has to offer [99, 125, 126]. *Record linkage* (RL) is a process of data integration which focuses on identifying records that belong to the same real-world entity either within the same data source or across different data sources [20, 38, 73, 94, 225]. RL allows more sophisticated data analytics which can lead to improved decision making, identifying terrorist activities, and recognising disease transmission, to name a few. In many RL applications, the entities that are being linked refer to people. For instance, in a RL project where two databases from a pharmaceutical company and a hospital are being linked, the records in those databases can refer to customers and patients, respectively [14].

In recent years, the demand for RL has increased significantly due to its ability to allow advanced data analytics as well as improve data quality and enrich data by resolving duplicate records [37, 38]. However, RL is a challenging task since there are often no unique entity identifiers available across the databases to be linked [38, 40]. Therefore, personally identifiable information such as names, addresses, telephone numbers, and dates of birth, are commonly being used to conduct RL.

Various privacy and security concerns emerge when such personal information has to be used and exchanged during the RL process. Most privacy and confidential policies enforced by both private and government organisations around the world limit the exchange of personally identifiable information across different parties [2, 221]. *Privacy-preserving record linkage* (PPRL) focuses on developing techniques that enable linkage of different databases using personally identifiable information while preserving the privacy and the confidentiality of the entities that are represented by the records in these databases [26, 176, 200].

Over the past two decades, a number of different PPRL techniques have been proposed [155, 176, 177, 201]. We discuss these PPRL techniques in detail in Chapter 2. In order to be used in a real-world linkage project, a PPRL technique should satisfy a set of requirements: (1) be scalable to link large databases across multiple parties, (2) have the ability to compute the approximate similarities [1] of records, (3) have the ability to assess linkage quality, and most importantly (4) provide strong privacy guarantees against privacy attacks. Hence, assessing the privacy guarantees provided by PPRL techniques is of utmost importance. One such assessment approach is to apply different privacy attacks on databases protected by a PPRL technique in order to identify the weaknesses of this technique. Exploring and developing attacks on different PPRL techniques ensures that those techniques can be securely applied in real-world applications. A number of attacks have been proposed to exploit the weaknesses of PPRL techniques by trying to reidentify encrypted and/or encoded sensitive values in databases [42, 119, 121]. However, as we further discuss in Section 1.3, existing privacy attacks on PPRL have several limitations including parameter constraints and significant memory and runtime requirements. Therefore, further research into novel attack methods (that overcome the limitations of existing attacks) is important to identify different vulnerabilities of PPRL techniques and how they can be applied in real-world linkage applications.

1.1.1 Importance of Information Privacy

Information privacy can be defined as a person's right to have control over their personal data, how they are being collected, and utilised [187]. *Personally identifiable information* (PII) is defined as information which can be used either alone or in combination with other information to distinguish or locate the identity of an individual [117]. A person's PII includes his/her name, address, date of birth, social security number, gender, health information, and so on. Ensuring the privacy of such information is important because a breach of an individual's privacy can affect his/her reputation, trust, social relationships, and more importantly, his or her own physical and emotional well-being [40].

With the rapid development of information technology and the absence of appropriate privacy-preserving techniques, data breaches occur on a regular basis [13, 221]. For example, in 2018 it was disclosed that the personally identifiable information of

[1] Approximate similarity: a normalised numerical similarity value between 0 and 1 that refers to the similarity between the two attribute values or records [38].

87 million Facebook users has been obtained allegedly without the consent from the actual users by a British consulting firm named Cambridge Analytica[2]. The 2013-14 data breaches at Yahoo are considered to be the largest data breaches in the history of the Internet, where 3 billion of Yahoo's user accounts were breached and compromised[3]. According to the disclosure made by Yahoo in 2016, not only people's names, email addresses, and telephone numbers were compromised but even their passwords, security questions, and the corresponding answers.

In 2016 a group of researchers from the university of Melbourne were able to successfully de-anonymise records in a supposedly anonymised medical billing record database which was shared publicly by the Australian Department of Health [47]. Data breaches at eBay, Uber, and Friend Finder are a few other examples which claimed to have compromised 145, 57, and 412 million user accounts, respectively[4]. These incidents stress the need for proper privacy-preserving methods as well as adequate rules and regulations to protect the personally identifiable and potentially sensitive information of people [40].

1.1.2 Data Privacy Policies and Laws

Many countries around the world have data privacy policies and laws put forth by their governments to govern proper usage of personal and sensitive information of people residing in those countries.

The European Union (EU) General Data Protection Regulation (GDPR)[5], which came into effect on 25 May 2018, enforces the privacy of all European citizens by requiring organisations to use the personally identifiable information of individuals in a lawful and fair way while ensuring the confidentiality of this information. The regulations incorporated in GDPR cover a wide range of information privacy aspects, including but not limited to, data breach notifications, data processing transparency, the right to be forgotten, individual pseudo-anonymity, and data portability.

The Australian government's recently established new data protection scheme, called the Notifiable Data Breaches (NDB)[6] scheme, which came into effect on 22 February 2018, states that in case of an *eligible data breach*, within 30 days an organisation must notify individuals whose personal information is likely to have been affected due to that data breach. Furthermore, the Australian government's Data Matching Act[7] implies that all agencies that carry out data matching tasks must comply with regulations of the Privacy Act 1988[8]. These legal restrictions are in place to mitigate the disclosure of personal and sensitive data and to prevent possible data breaches. The 13 principles of Australia's Privacy Principles (APPs)[9] govern the stan-

[2] https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal
[3] https://en.wikipedia.org/wiki/Yahoo!_data_breaches
[4] https://en.wikipedia.org/wiki/List_of_data_breaches
[5] https://gdpr-info.eu/
[6] https://www.oaic.gov.au/privacy-law/privacy-act/notifiable-data-breaches-scheme
[7] https://www.oaic.gov.au/privacy/other-legislation/government-data-matching
[8] https://www.oaic.gov.au/privacy/the-privacy-act/
[9] https://www.oaic.gov.au/privacy/australian-privacy-principles/

dards, rights, and obligations around the collection, use, and disclosure of sensitive information, the integrity and correction of personally identifiable information, and the rights of individuals to access their personally identifiable information.

The United States' Health Insurance Portability and Accountability Act of 1996 (HIPAA)[10] is aimed at protecting sensitive patient health information from being disclosed without the patient's consent or knowledge. To comply with HIPAA, all organisations that are covered by HIPAA must ensure the confidentiality, integrity, and availability of all electronic health information, detect and safeguard against anticipated security threats, and potential unauthorised disclosure of information.

The personal Information Protection and Electronic Documents Act (PIPEDA)[11] in Canada governs how organisations can collect, store, and utilise personally identifiable information about individuals. According to the act, organisations must obtain consent when collecting and/or disclosing individuals' personally identifiable information. Furthermore, people have the right to access their personally identifiable information held by an organisation and have the right to challenge its accuracy.

1.2 Motivation of the Research

As we discussed in the previous section, due to the lack of unique entity identifiers across databases, RL requires personally identifiable information, also known as quasi-identifying (QID) values, of entities for the linkage of records. However, utilising such personally identifiable information leads to different privacy and security concerns [40]. PPRL, on the other hand, aims to develop methods that allow accurate linkage of records using encoding and/or encryption techniques in such ways that the privacy of the entities that are represented by the records is preserved.

In PPRL, the QID values of records that are being linked are encoded (masked), perturbed, or converted into hash codes, such that the original (plaintext) values of those attributes are obscured. However, at the same time, the encoding and/or encryption technique used for PPRL must ensure that the original similarities between records are not altered, so that the linkage of those records can be performed accurately to identify matching record pairs across databases. As we discuss in Chapter 2, one popular binary representation based encoding technique used in PPRL is Bloom filters (BF) encoding [177]. Due to its ability to allow efficient and approximate matching of records, BF encoding is now being used in several real-world PPRL applications [10, 29, 159]. However, recent research has demonstrated that BF encoding is susceptible to different privacy attacks, as we discuss in Chapter 4.

Over the past decade, a variety of attack methods have been proposed for different PPRL techniques to reidentify encoded attribute values using various techniques [42, 119, 121, 147]. However, these privacy attacks have several limitations:

[10] https://www.cdc.gov/phlp/publications/topic/hipaa.html

[11] https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information\-protection-and-electronic-documents-act-pipeda/

1. They require an adversary to obtain knowledge of specific parameters employed in the PPRL encoding process. For instance, some existing attacks on PPRL will only work when the number of hash functions or the hashing method used for BF encoding is known in advance [119, 121, 147].

2. They are only applicable on a single encoding technique, or only on limited sets of parameter values. For instance, the privacy attack on BF encoding proposed by Kroll and Steinmetzer [119] can only be applied if the BF encoding is based on double hashing [177], as we discuss in detail in Section 4.2.

3. They have significant memory and runtime requirements which makes such attacks infeasible on large encoded databases. For instance, the privacy attack proposed by Niedermeyer et al. [147] took several days to run on a dataset containing 10,000 values.

In Chapter 4 we discuss in more details the functionalities of existing attacks and their drawbacks. Due to these limitations, most existing attacks are infeasible to be performed in various real-world PPRL scenarios to identify privacy issues in PPRL encoding techniques. Furthermore, since it has already been shown that PPRL techniques, especially BF encoding, can be attacked to reidentify encoded sensitive values [42, 119, 121], further research to develop novel attack methods is important to explore and identify the vulnerabilities and weaknesses of these techniques compared to the privacy protection they provide. Based on these research gaps, we identified three challenges that are related to privacy attacks on PPRL:

1. **Challenge 1**: Categorising existing privacy attacks on PPRL techniques based on the characteristics of those attacks such as their underlying assumptions, employed techniques, exploited vulnerabilities, and evaluation criteria.

2. **Challenge 2**: Quantifying the vulnerabilities of both plaintext and encoded values in a sensitive database. In PPRL such a vulnerability assessment is challenging because different plaintext databases can have different vulnerabilities and different PPRL techniques output encodings with diverse characteristics.

3. **Challenge 3**: Exploring and exploiting new vulnerabilities in PPRL techniques by developing more efficient privacy attacks.

Solving these challenges will eventually lead to the development of better PPRL techniques with improved privacy. Benefits of being able to use such techniques in real-world applications motivated us to conduct this research study.

1.3 Aims of the Research

The aims of this thesis are to *explore the means of quantifying the vulnerabilities that exist in PPRL encoding techniques, and investigate attack methods for PPRL techniques with the purpose of identifying the limitations of those techniques; and furthermore to evaluate these*

novel attack methods in terms of their accuracy in reidentifying encoded attribute values and/or entities and their scalability to large real-world databases.

In the process of achieving these aims, the following objectives are considered.

- **A comprehensive analysis of existing privacy attacks on PPRL**
  To the best of our knowledge no study has so far been conducted to categorise privacy attacks on PPRL. Each existing attack method has been evaluated by its authors using different databases with different parameter combinations, which makes them difficult to compare against each other. Our objective is to propose a framework to categorise and evaluate existing attacks on PPRL because such a framework is an important step towards comparing attack methods and identifying their limitations, as we discuss in Chapter 3.

- **Explore and measure vulnerabilities of sensitive data and PPRL encoding techniques**
  In the context of statistical disclosure control [59, 193], a number of techniques have been proposed to measure the disclosure risks of anonymised data. However, in PPRL, no systematic analysis has so far been proposed to quantify the vulnerabilities associated with different PPRL encoding techniques. Our objective is to investigate and quantify the vulnerabilities in terms of both plaintext and encoded databases.

- **Develop attack methods which systematically explore the vulnerabilities of current PPRL encoding techniques**
  Existing privacy attacks exploit vulnerabilities of different PPRL techniques utilising methods such as frequency alignments [119], bit pattern analysis [42], and constraint satisfaction [121]. However, there still remain unexplored vulnerabilities in existing PPRL techniques in terms of privacy attacks. In this objective we develop more sophisticated attacks on existing PPRL techniques to find novel vulnerabilities in those techniques that could lead to the reidentification of encoded personally identifiable values.

- **Develop attack methods that are independent of the PPRL encoding techniques and/or the knowledge about parameter settings used for the encoding**
  Most existing attack methods depend on certain assumptions such as the adversary's knowledge of the parameters used in the encoding process. Additionally, some attacks only work on a single encoding technique or values which have been encoded using certain combination of functions, such as double hashing [147] or single attribute encoding [42] as used in BF encoding. Such assumptions limit the feasibility of an attack to be applied in real-world PPRL applications. We aim to develop attacks which are independent of the PPRL encoding techniques and/or parameters used so that they can be applied successfully in a wide range of real-world scenarios.

- **Develop attack methods that are scalable to large databases**
  When developing PPRL attack methods, one of the main challenges that an ad-

versary might face is the excessive time and memory consumption of an attack which makes it less applicable on large databases. We develop attack methods which can achieve high efficiency while achieving high reidentification accuracy. Accomplishing this objective is important when conducting privacy attacks in any real-world applications with large databases.

1.4 Contributions of the Research

Our research consists of five main contributions which are aimed at addressing the challenges discussed in Section 1.2. These contributions are also based on the aim and objectives of our research described in the previous section.

1. **A taxonomy of existing attack methods on PPRL techniques**
   As we discuss in Chapter 3, we propose a taxonomy to categorise existing attack methods on PPRL techniques based on their characteristics. We consider three main aspects (adversarial, technical, and practical) to analyse and categorise attack methods.

2. **A vulnerability framework for PPRL**
   It is important to measure the vulnerabilities of both plaintext and encoded databases before they are being used in a PPRL project in order to understand how the corresponding values in these databases can possibly be attacked. As we discuss in Chapter 5, we propose an evaluation framework for PPRL that can quantify the vulnerabilities of plaintext as well as values that have been encoded using different PPRL techniques.

3. **A frequency based privacy attack on multiple match-key encoding**
   As we discuss in Chapter 6, we develop an attack based on frequency analysis on a recently proposed PPRL encoding technique named multiple match-keys (MMK) [163]. This attack aligns frequent plaintext value combinations (called plaintext match-keys) with frequent hash codes (called encoded match-keys) from an encoded database to reidentify encoded QID values. This attack is the first to explore the vulnerabilities of MMK encoding, where we also propose a set of recommendations in Chapter 6 which can be used to strengthen this encoding technique to prevent our attack.

4. **A pattern-mining based cryptanalysis attack on Bloom filter encoding**
   As we discuss in Chapter 7, we extend a privacy attack on BF encoding initially proposed by Christen et al. [43] to improve its reidentification accuracy. This extended attack utilises a maximal frequent pattern-mining approach, a language model, and a partitioning technique to identify co-occurring bit positions in BFs that can encode frequent q-grams. Unlike other frequency attacks on BF encoding, this attack does not require frequent BFs and plaintext values and it is able to reidentify QID values encoded in BFs even if all the BFs in the encoded database are unique.

5. **A graph matching based privacy attack which is independent of the used PPRL encoding technique**
   As we discuss in Chapter 8, we develop a similarity attack which can be applied on any PPRL encoding technique that calculates approximate similarities between encoded records. In this attack we use a graph matching based approach to align nodes between two similarity graphs generated using the plaintext and encoded databases. The attack uses a set of node neighbourhood based features to find the correct alignments between nodes, and we apply this attack on three different PPRL encoding techniques to show its applicability on real linkage applications.

The results of our research will lay foundations to better understand the weaknesses and vulnerabilities of existing PPRL techniques, to explore how to strengthen the privacy guarantees of those techniques, and to develop novel PPRL techniques that are resilient against privacy attacks.

1.5 Scope of the Research

The scope of our research study is based on the following aspects:

- **Scope of the privacy attacks**: Our research is focused on investigating attacks specific to perturbation based encoding techniques, such as BF encoding [177], because of their popularity in real-world PPRL applications [10, 29, 153]. Other PPRL techniques such as embedding space [26, 173] and secure multi-party computation based encoding and/or encryption techniques are out of the scope of our research study [96, 109, 123]. We also propose our attacks only for textual based PPRL encoding techniques because textual based personally identifiable information are more commonly used in PPRL applications [26, 29, 40]. Limited research has been conducted for the encoding of other data types such as numerical, date, and geocoding values [106, 107, 203]. Developing attacks on encoding techniques for other data types is a potential future work of our research, as we discuss in Chapter 10.

- **Assumptions on adversary's knowledge and access to data**: When designing different attack methods it is imperative that certain assumptions be made. With each proposed PPRL attack, as we describe in Chapters 6 to 8, we make a set of assumptions on the prior knowledge of the adversary and the availability of plaintext databases to the adversary. One such commonly made assumption is that the adversary has access to a publicly available plaintext database [42, 119, 121]. In this research study, we do not focus on developing attack methods where the adversary only has access to an encoded database. However, we theoretically discuss the possibility of such ciphertext-only attacks in the context of PPRL in Chapter 3.

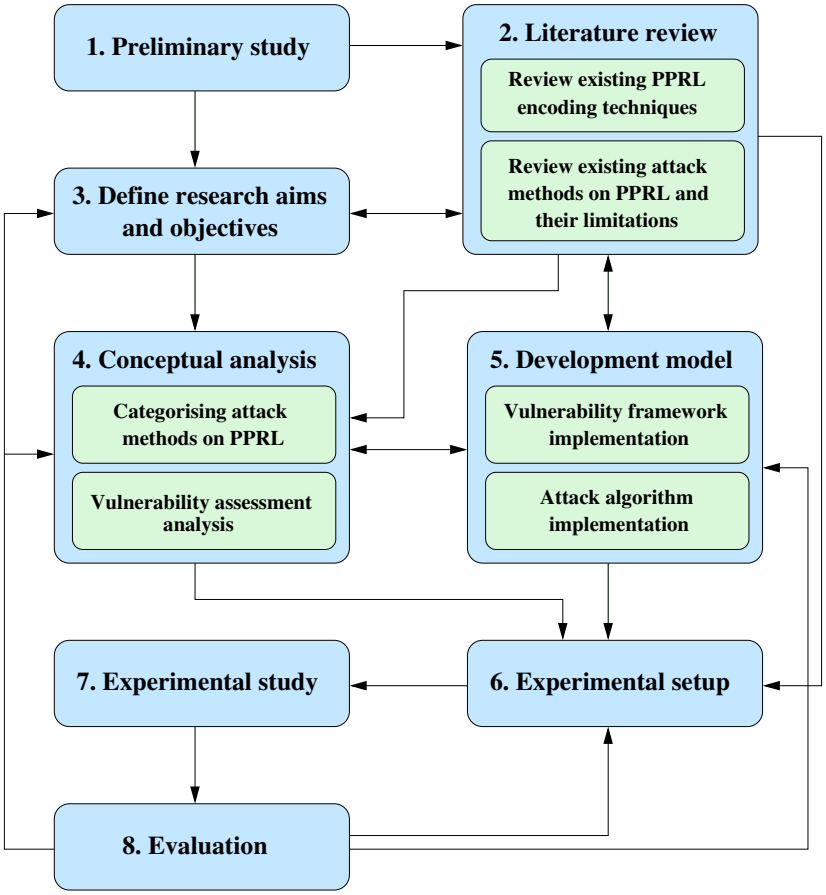


Figure 1.1: The proposed research methodology.

- **Experimental evaluation**: We used two real-world databases, a North Carolina voter registration (NCVR) database[12] and a Michigan voter registration (MVR) database[13], and two synthetic European (EURO) databases[14], where one is a census (CS) database and the second is a customer information system (CIS) database, for the experimental evaluation of the proposed privacy attacks and the vulnerability framework. These databases have been selected because they contain attributes (personally identifiable information) that are commonly used in PPRL research for the encoding of records and they are publicly available. We use one set of databases for the experimental evaluations conducted in individual Chapters 5 to 8, and another set of databases for the comparative evaluation conducted in Chapter 9. We discuss the characteristics of these selected databases in more detail in their relevant chapters.

[12]https://dl.ncsbe.gov/
[13]http://michiganvoters.info/
[14]https://ec.europa.eu/eurostat/cros/content/job-training_en

1.6 Research Methodology

As discussed in Section 1.3, the purpose of this thesis is to design a vulnerability assessment framework for sensitive databases and to develop attack methods to evaluate the privacy guarantees of different PPRL techniques. The methodology we followed in achieving the aims and objectives of our research, as we illustrate in Figure 1.1, is as follows:

1. **Preliminary study**: In this step we obtained the necessary background knowledge about RL and various PPRL techniques by studying the relevant literature and analysing existing techniques to find potential research gaps.

2. **Literature review**: In this step we conducted a comprehensive analysis into existing PPRL techniques and attack methods on those techniques to investigate weaknesses of PPRL techniques, and to identify the limitations of existing attack methods. We also studied other research areas such as pattern-mining, cryptography, and graph matching, that are related to our research study.

3. **Define research aims and objectives**: Referring to the identified gaps in research on existing PPRL techniques and attack methods, we defined our research aims and relevant objectives. We also outlined the scope of our research study under this step.

4. **Conceptual analysis**: In this step we first conceptually analysed existing attacks on PPRL techniques with the purpose of categorising them based on their characteristics. Next, we designed a framework to evaluate the vulnerabilities of sensitive and encoded data as used in PPRL.

5. **Development model**: Here we first implemented the designed vulnerability analysis framework as a proof of concept. Next, we designed and implemented three different attack methods on multiple PPRL techniques.

6. **Experimental setup**: Here we designed the experiments to be run by selecting suitable databases, parameter settings, evaluation metrics, and relevant baselines for comparisons. We refer to the literature study we conducted on PPRL techniques and previous attack methods on those techniques when setting up those experiments.

7. **Experimental study**: In this step we executed the designed experiments and obtained results for evaluation.

8. **Evaluation**: In this step we compared the collected results of our proposed three attack methods with selected baselines, and evaluated these methods in terms of their accuracy, robustness, and scalability. Based on the obtained experimental results we also described the insights gained, and discussed potential future research directions.

1.7 Thesis Outline

The remaining chapters of the thesis are structured as follows. In Chapter 2 we discuss the necessary background about the overall process of privacy-preserving record linkage (PPRL) and its evaluation and privacy aspects. In Chapter 3 we then propose our taxonomy of attacks on PPRL that we use to categorise existing attacks on different PPRL encoding techniques. In Chapter 4 we provide a detailed discussion of existing attacks on PPRL, including their limitations. In Chapter 5 we then propose a framework for PPRL that can quantify the vulnerabilities in both the plaintext and encoded databases. In Chapters 6, 7, and 8 we propose three novel privacy attacks: a frequency based attack, a pattern-mining based attack, and a graph matching attack, on different PPRL techniques and evaluate those attacks using real-world and synthetic databases. In Chapter 9 we then conduct a comparative evaluation between the proposed three attack methods and several other baseline attacks. Finally, in Chapter 10 we conclude our research study, summarise the achieved results, and discuss directions for future work of this research domain.

We include two appendices to provide the reader additional material to better understand the content in this thesis. In Appendix A we extend our discussion about Bloom filter (BF) encoding for PPRL where we describe various aspects such as BF hashing techniques, encoding techniques, and hardening techniques. In Appendix B we discuss the background on assessing disclosure risk in microdata.

Chapter 2

Background

This chapter details the background on record linkage (RL) and privacy-preserving record linkage (PPRL) including their basic concepts, principles, and techniques. In Sections 2.1 and 2.2 we first provide brief introductions to RL and PPRL, respectively. In Section 2.3 we then discuss the process of PPRL in detail and in Section 2.4 we detail the privacy aspects of PPRL. In Section 2.5 we discuss existing PPRL techniques proposed over the years and in Section 2.6 we outline different PPRL evaluation metrics. Finally, in Section 2.7 we summarise the content described in the chapter.

2.1 Record Linkage (RL)

The linking of records across databases has been a requirement in various domains such as census and public health data analytics for many decades due to the benefits it yields [38]. Social scientists have been linking historical census data to analyse social and geographic mobility in certain communities [168], and health records of individual patients are commonly linked together using their personal information such as names and dates of birth to provide better health services [38, 40, 223]. The term *Record Linkage* (RL), was first coined by Halbert Dunn [60] in 1946. According to Dunn, each unique individual in the world creates a *book of life* which starts with a birth record and ends with a death record. Dunn describes RL as the process of assembling pages of this book into a volume. He explains the importance of RL to both individuals and organisations as a tool to produce statistics, obtain complete and accurate records, and certify certain records as a proof of identification [60].

After Dunn, the idea of automating the RL process using computers was proposed by Howard Newcombe and his colleagues [146] in 1959, taking the initial step towards *probabilistic record linkage*. During a process of linking marriage records of people with their birth records, Newcombe and colleagues have used the Soundex phonetic method [169] to encode last name values in records in order to reduce the inconsistencies and variations of personal names. This work was later formalised into a mathematical model by Ivan Fellegi and Alan Sunter [73] who developed a theory for probabilistic RL. Their work was the first to prove that for a given comparison space, an optimal linkage rule can be defined under the assumption that the attributes used for the linkage are conditionally independent. Numerous research

and applications of RL have been inspired by this pioneering work of Fellegi and Sunter ever since [224, 225, 226].

RL can be defined as the problem of classifying pairs of records from either a single or multiple databases into two classes, *matches* and *non-matches* [38, 200]. The class of *matches* contains pairs of records that are assumed to refer to the same real-world entity, while the class of *non-matches* contains pairs of records where the two records correspond to different real-world entities. In some linkage protocols, there can also be a third class named *potential-matches* where a manual clerical review is required to classify a given record pair as a match or a non-match [37, 84, 157].

Due to the absence of unique entity identifiers across databases, such as social security numbers (SSNs), RL is often based on common personally identifiable information known as quasi-identifiers (QIDs) [38, 40]. For instance, if the databases contain information about people, then the QIDs can be their names, addresses, dates of births, ans so on.

In practice, linking of records using QIDs involves of two main challenges, *scalability* and *linkage quality*.

- **Scalability**: In a naïve linkage scenario of two databases, each record in one database needs to be compared with every record in the other database making the comparison space to grow quadratically in terms of the sizes of the databases [38]. Furthermore, the complexity grows exponentially with the increase of the number of databases to be linked [40]. For instance, a linkage of three databases each with 100 records will result in 1,000,000 record pair comparisons. This makes the naïve pairwise comparison process difficult to scale to large databases and increasing the number of participating databases. Therefore, to improve the scalability blocking techniques are commonly used. As we discuss further in Section 2.2, blocking reduces the comparison space by generating blocks of records which contain records that are more likely to refer to matches [5, 37, 104, 159].

- **Linkage quality**: Real-world data often contain errors, variations, missing, and outdated values [88, 165]. These inconsistencies in data will have an impact on linkage quality since they can increase the number of false positives (records incorrectly identified as belonging to the same entity) and false negatives (records incorrectly identified as not belonging to the same entity). Therefore, it is vital to have linkage techniques that are capable of comparing records accurately even when the attribute values contains errors and variations [38].

2.2 Privacy-Preserving Record Linkage (PPRL)

As we discussed in Section 2.1, RL is often based on QID values of entities such as the names and addresses of people. However, regulatory ethical and legal obligations can limit the use of such personally identifiable information in the linkage process in order to protect the privacy and confidentiality of entities [40, 87, 210]. As we

outlined in Section 1.1.2, governments around the world have enforced various data privacy laws and regulations to prevent potential privacy breaches. Therefore, the requirement of databases to be linked in such a way that no sensitive information about the entities is being revealed (unless otherwise indicated), has recently gained significant attention [40, 148]. This has led to a novel research area called *privacy-preserving record linkage* (PPRL) [26, 40, 87, 211], which is also known as *private record linkage* (PRL) [64, 208, 220, 228], or *blind-folded record linkage* [45, 219].

PPRL aims to develop techniques that enable the linkage of records in one or more databases without revealing any sensitive or confidential information about the entities that are represented by these records to parties both internal as well as external to the linkage process (as we discuss in Section 2.4). PPRL can be formally defined [40, 209, 213] as follows:

Definition 2.2.1. Privacy-preserving record linkage (PPRL)
Let us assume d database owners DO_1, DO_2, ..., DO_d each with a sensitive database, $\mathbf{D}_1^s$, $\mathbf{D}_2^s$, ..., $\mathbf{D}_d^s$, respectively, where $d > 1$, that need to be linked. Privacy-preserving record linkage aims to identify which record pairs (r_a, r_b), where $r_a \in \mathbf{D}_i^s$, $r_b \in \mathbf{D}_j^s$, $1 \leq a \leq |\mathbf{D}_i^s|$, $1 \leq b \leq |\mathbf{D}_j^s|$, $1 \leq i, j \leq d$, and $i \neq j$, represent the same real-world entity by comparing the personally identifiable information in records without disclosing any sensitive information about these entities during the process. At the end of the PPRL process each DO only learns which records in its sensitive database have been classified as matches with records in the other databases. A DO does not learn actual values of the matched records in any of the databases of the other DOs.

Conceptually PPRL can also be viewed as an approximate secure set intersection operation between two or more databases where it discovers all common records between those databases. The each DO will only learn records in its own database that belong to the intersection.

Apart from the two challenges of RL we discussed in the previous section, *privacy* is the third major challenge of PPRL. Techniques employed in a linkage scenario should make sure that they provide high linkage quality while providing an adequate level of privacy for the entities in databases, whose records are being linked, against any existing privacy attacks. However, as we discuss in Section 2.5, PPRL techniques generally have a trade-off between scalability, linkage quality, and privacy.

Over the past two decades various PPRL techniques have been proposed to securely link records between databases by encoding and/or encrypting sensitive values. In general different privacy methods need to be utilised in different steps of the PPRL process in order to preserve the privacy of the entities in the databases to be linked, as we describe next.

2.3 The PPRL Process

A PPRL technique needs to be able to link true matching record pairs across different databases correctly and efficiently while ensuring the privacy of the entities

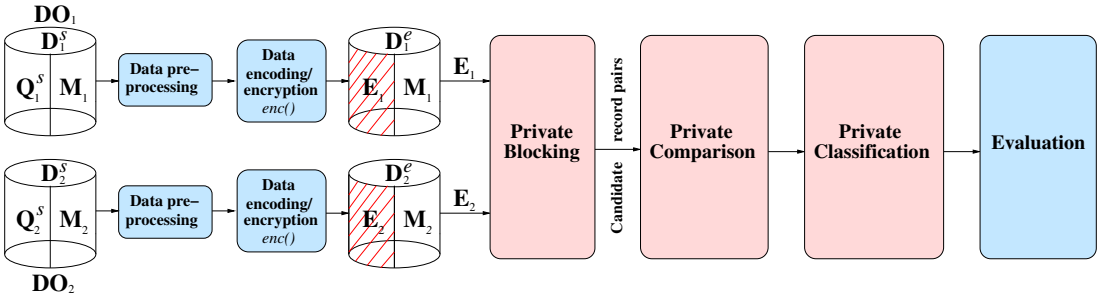


Figure 2.1: Main steps of the PPRL process assuming two databases $\mathbf{D}_1^s$ and $\mathbf{D}_2^s$ from database owners (DOs), $\mathbf{DO}_1$ and $\mathbf{DO}_2$, are being linked. As can be seen, the DOs first conduct the data pre-processing and encoding/encrypting on their own databases separately, and then the encoded/encrypted QID values $\mathbf{E}_1$ and $\mathbf{E}_2$ of the encoded/encrypted databases $\mathbf{D}_1^e$ and $\mathbf{D}_2^e$ are being linked using private blocking, private comparison, and private classification steps. In the final step the obtained results are evaluated. Here, $\mathbf{Q}_1^s$ and $\mathbf{Q}_2^s$, and $\mathbf{M}_1$ and $\mathbf{M}_2$ represent the plaintext QID values and the microdata of the corresponding sensitive databases, respectively, and *encode*() represents an encoding/encryption function.

represented by these records. An adversary (internal or external, as we describe in Section 3.5.1) must not be able to infer any sensitive information throughout the entire linkage process or afterwards. We now discuss how these privacy requirements can be achieved in each step of the PPRL process.

The general PPRL process consists of five main steps [209], as illustrated in Figure 2.1: data pre-processing and encoding/encryption, private blocking, private comparison, private classification, and evaluation. For improved readability, in the following we refer to both encoding and encryption as encoding.

The quality of any linkage depends on the accuracy, completeness, and consistency of the data, while in real-world scenarios data often contain various errors and variations including name variations and nicknames, spelling mistakes, missing values, and use of abbreviations. Data pre-processing includes cleaning, normalisation, and standardisation of data before a linkage to minimise the effects of such errors and variations. This first step of the PPRL process does not require any privacy technique since it can be conducted independently by each database owner (DO) [38]. However, to maintain consistency, DOs should agree upon a set of pre-processing steps at the beginning of the PPRL process [45, 165].

After the databases have been cleaned, each DO encodes its own database using an agreed encoding technique [212]. In order to encode the sensitive values in their databases, the DOs first need to agree upon a set of QID attributes common across all the databases to be used, and an encoding function *encode*() with a common set of parameters $\mathbf{p}$ and a secret key s to be used for the encoding. For instance, the set of QIDs can be {*FirstName, LastName, StreetAddress, City, Zipcode*} and *encode*() can be Bloom filter encoding with its corresponding set of parameters (as we describe in Section 2.5.1). In the following we describe the encoding process in more detail.

We assume a sensitive database $\mathbf{D}^s = (\mathbf{Q}^s, \mathbf{M})$, where $\mathbf{Q}^s$ is the list of actual attribute values of records in the selected QIDs, and $\mathbf{M}$ are the microdata in $\mathbf{D}^s$ (such as medical or financial details of individuals) that can be used in the analysis. Each DO_i encodes the values in $\mathbf{Q}^s$ in its database $\mathbf{D}^s_i$ using the encoding function $encode()$, where $1 \leq i \leq d$ and d is the number of databases to be linked. We denote the encoding of a database $\mathbf{D}^s_i$ using its list of QID values $\mathbf{Q}^s_i$ as:

$$\mathbf{E}_i = encode(\mathbf{Q}^s_i, \mathbf{p}, s), \tag{2.1}$$

where s is the secret key used for the encoding, and $\mathbf{E}_i$ is the list of encoded QID values. We represent an encoded database as $\mathbf{D}^e = (\mathbf{E}, \mathbf{M})$ and $|\mathbf{E}| = |\mathbf{Q}^s|$ where $|\cdot|$ represents the size of a database (its number of records). The microdata, $\mathbf{M}$, are not encoded because these are the values of interest to a data consumer (as we define in Section 2.4) after the linkage process. However, $\mathbf{M}$ can also include certain QID values, such as gender, year of birth, and/or zipcode, that might be useful in an analysis task. In such a scenario those QID values are often generalised to prevent reidentification before being published or used for an analysis [192]. After the encoding/encryption process, each database $\mathbf{D}^e_i$ from DO_i is linked using the corresponding encoded QID values $\mathbf{E}_i$.

In the private blocking step the encoded records which likely correspond to matches are grouped into blocks. As we described in the previous section, one of the main challenges in both RL and PPRL is its quadratic complexity [40]. This becomes even more challenging with multi-party PPRL [157, 206], where the number of databases to be linked increases. The aim of private blocking is to reduce the comparison space by grouping records which are likely to correspond to matches into blocks [157], while ensuring that the privacy of the entities is being preserved. Blocking in PPRL is performed on encoded values, where various methods including phonetic codes, clustering, and locality sensitive hashing have been used for the blocking of databases [103, 159, 160].

The private comparison step focuses on calculating the similarities between encoded values of record pairs that are in the same blocks generated in the private blocking step [27, 64, 102]. Private comparison should also ensure that no sensitive information about the entities is being revealed while allowing accurate comparison of records. Encoded values can be compared either using exact or approximate comparison functions. Since encoded values can be represented by different data structures, such as hash-tables and Bloom filters, different comparison functions need to be used to compare such encoded values [102, 177]. These comparison functions include secure Levenshtein edit distance, set based Dice coefficient and Jaccard coefficient, and so on [14, 62, 102, 158, 177].

Next, in the private classification step, the compared record pairs are generally classified into two classes, *matches* and *non-matches* (as we discussed in Section 2.1), based on the calculated similarities between them [38]. Threshold-based [38], probabilistic [73, 224], cost-based [213], rule-based [145], and machine learning-based [22, 36, 76] classification techniques are commonly used in PPRL applications. Similar to

private blocking and private comparison, private classification needs to be conducted in a way that no sensitive information of the entities being linked is disclosed. The linkage results (pairs of record identifiers from the class *matches*) are then shared between the corresponding DOs allowing them to conduct further analysis. Alternatively, linkage results can also be shared with a data consumer (DC), as we discuss in Section 2.4.

At the end of the PPRL process the DOs will only learn which records in their own sensitive database $\mathbf{D}_i^s$ have been matched with records in the other sensitive databases. If a third party is employed in the linkage process (such as a linkage unit (LU) as we discuss in Section 2.4), the linkage protocol should ensure that the third party will learn nothing about the sensitive values in any of the $\mathbf{D}_i^e$.

The final step of PPRL is evaluating the linkage process in terms of the quality of the linkage results, scalability of the linkage to large databases, and the privacy protection offered by a linkage protocol [157, 200]. This is a challenging task, especially when evaluating linkage quality in a real-world linkage scenario, because using ground truth data in a PPRL context could reveal private information about the entities involved in the linkage [209]. We discuss different evaluation metrics used in PPRL in Section 2.6.

2.4 Privacy Aspects of PPRL

In PPRL applications, different privacy aspects needs to be considered in order to preserve the privacy of the entities that are being linked. These aspects include the roles of parties, the number of parties, and different adversary models.

2.4.1 The Roles of Parties

In a PPRL protocol, different parties with different responsibilities can participate in the protocol. These parties can be either internal or external to the protocol and they can be categorised as follows.

- **Database owner (DO)**: A person or an organisation who holds a database that is to be linked [40]. DOs are also referred to as data owners or data custodians [200]. Depending on the PPRL protocol used, DOs can undertake certain operations in the linkage process including pre-processing and encoding of their own databases [209]. Healthcare service providers, business owners, and government departments are some examples of DOs.

- **Linkage unit (LU)**: A party that participates in the linkage process to conduct the linkage using the encoded QID values it receives from the DOs [210]. A LU can either be one of the DOs themselves or a party external to DOs [40, 64, 209]. If an external LU is employed, the encoded QID values $\mathbf{E}_i$ of each DO_i are sent to the LU to conduct the linkage of the databases [35, 64, 209]. The LU then conducts the linkage of encoded records and returns the results to the

corresponding DOs. In general the LU does not have access to the plaintext QID values $\mathbf{Q}^s$ or microdata $\mathbf{M}$ of the DOs, and does not know any of the parameter settings $\mathbf{p}$ and secret key s used in the encoding process [164].

- **Data consumer (DC)**: A person or an organisation who utilises the linked data at the end of the PPRL process to conduct data analysis [23, 40]. Depending on the requirements of (or agreements made prior to) a PPRL process, a DC will have access to the microdata $\mathbf{M}$ from the sensitive database $\mathbf{D}^s$ and a set of unique identifiers (which may or may not be generated randomly) assigned to matched records, where $\mathbf{M}$ needs to be properly anonymised before they are being made available to a DC to prevent reidentification of entities represented by $\mathbf{M}$ [54, 192]. A DC can either be one of the DOs whose databases have been linked, or an external party such as a researcher.

Depending on the protocol employed, several other parties can also be involved in a linkage process [40]. Such parties include a global authority (a party which plays the role of a central, public, and trusted regulatory agency who creates a global summary using local summaries), a facilitator (a party which helps the DOs to negotiate and plan their common objectives of a linkage process), and a data controller (a party who incorporates relevant laws and regulations for a linkage protocol and who provides permissions to the parties involved in the protocol).

2.4.2 Number of Parties

PPRL protocols can be categorised based on the number of parties involved in the linkage process.

- **Two-Party Protocols**: In a two-party protocol only two DOs participate in the PPRL process where they collaborate with each other to perform the linkage of their sensitive databases [33, 35, 164, 201, 212]. These protocols are designed as a more secure approach than the protocols that require an external party (as we discuss below). A two-party protocol will prevent any type of collusion that might happen between the DOs and an external party. A collusion in PPRL is an agreement (kept hidden from the other parties) or cooperation between two or more parties to work together in order to learn sensitive information about the other parties who participate in the linkage protocol [40]. However, these protocols often require more computationally complex algorithms to ensure the privacy of sensitive information, such as secure set intersection and secure scalar product [220]. On the other hand, the two-party protocols also generally require higher computational and communication costs due to the use of advanced encoding or encryption mechanisms such as homomorphic encryption [109].
- **Three-Party Protocols**: In a three-party protocol two DOs conduct the linkage of their sensitive databases employing a LU [6, 164, 174, 212]. The two DOs

first conduct the initial data pre-processing and encoding steps of their own databases, and then send the encoded data to the LU. The LU can be either a trusted or a semi-trusted (as we discuss in the following section) party who performs the linkage of the databases it receives from the DOs by comparing and classifying encoded record pairs. The record identifiers of resulting matched pairs are sent back to each DO after the linkage process. Depending on the agreed requirements, further details of matched records, such as the microdata **M** of matched pairs, can also be shared among DOs. However, in three-party protocols, a semi-trusted LU can collude with one of the DOs to infer information about the other the DO's sensitive database. Moreover, the LU can perform an attack on encoded QID values it receives from DOs (assuming the LU knows some auxiliary information about the data that is being linked) using publicly available plaintext data and other available information such as frequency distributions of encoded values [42, 119, 121].

- **Multi-Party Protocols**: In a multi-party protocol more than two DOs link their sensitive databases [149, 157, 200]. A LU may or may not be employed in a multi-party protocol [124, 205]. When a LU is employed, the protocol proceeds similar to a three-party protocol where the DOs exchange information about the encoding process, pre-process and encode their own sensitive data, and then send their encoded data to the LU to conduct the linkage of records [157]. However, the privacy techniques used for the linkage need to be scalable and efficient if multiple (more than two) databases are being linked. If, on the other hand, a LU is not employed in multi-party protocol, then the DOs conduct the linkage of records collaboratively by sharing encoded data among themselves [97, 124, 159]. Such a protocol provides more privacy protection compared to protocols with a LU since there is no possibility of a collusion between the DOs and the LU. However, these protocols generally require more sophisticated computational and communication techniques to allow efficient, scalable, and secure linkage of records [207].

2.4.3 Adversary Models

In the context of PPRL, four different types of adversary models can be considered: fully trusted (no-adversary), honest-but-curious, malicious, and covert. The following models discuss the behaviours of parties involve in a linkage protocol. However, parties outside the protocol cannot be fully trusted and by definition an external adversary always needs to be assumed as malicious.

- **Fully Trusted (No-adversary)**: In a fully trusted adversary model all parties involved in a linkage protocol follow all protocol steps without attempting to learn any information about any other party's sensitive data [135]. Furthermore, a fully trusted party does not provide invalid or fake data (instead of valid and correct data) to compromise the integrity of the linkage protocol.

Therefore, in practice, a this model can be regarded as a model without a potential internal adversary, as we discuss further in Section 3.5.1. However, in real-world scenarios not every party involved in a linkage situation should be treated as fully trusted as it is possible that they may try to learn sensitive information from the data and linkage results they receive from other parties.

- **Honest-But-Curious (HBC)**: In the HBC (also known as the semi-trusted) adversary model all parties follow the defined steps of a linkage protocol without deviating from them, while attempting to infer as much information as possible about the sensitive data of other parties [40, 135]. For instance, either the DOs or a LU can use the data they receive from the other parties (including the linkage results) to conduct certain explorations, such as frequency alignments or cryptanalysis attacks [42, 119]. Therefore, for a linkage protocol to be considered as secure under the HBC adversary model, the participants of the protocol must not be able to learn any sensitive information of any other parties' data at any stage. However, as we discuss in Section 3.5, collusion is possible under the HBC model, where several parties in the protocol can work together to infer sensitive information about the sensitive data of another party [162].

- **Malicious**: Under the malicious adversary model, the parties can behave maliciously by not following the necessary steps of the linkage protocol [87, 135, 141]. A malicious party can send invalid or falsified data to other parties, change agreed parameter values, and even refuse to participate in the protocol. Therefore, in the malicious adversary model, the privacy techniques used in a linkage protocol must guarantee that a malicious party will not be able to learn anything from the information it receives from any other party. Compared to the HBC model, achieving privacy under the malicious adversary model is more difficult since there are numerous ways for a party to deviate from the defined steps of a protocol. Existing PPRL techniques developed under the malicious adversary model [131, 141, 231] generally use secure multi-party computation (SMC) based techniques [135, 229] to provide improved privacy guarantees. These PPRL techniques also aim to make sure that even if a participating party becomes malicious the protocol will still be able to calculate the correct linkage results. However, such PPRL techniques generally incur high computation and communication costs compared to the linkage protocols that follow the HBC model, which make such PPRL techniques (currently) not feasible in real-world large-scale linkage applications.

- **Covert**: In practice, HBC based linkage protocols generally do not provide enough security, whereas the malicious model provides improved security generally at the cost of higher computational and communication complexities. The covert adversary model lies between the HBC and the malicious models, where it is suitable to be used in real-world linkage scenarios [12, 135]. In the covert adversary model it is assumed that the parties can behave maliciously and attempt to learn information about the sensitive data of other parties until

they are caught. The model assumes that an honest party in a linkage protocol can identify a cheating party with a certain probability (however, this probability will not be close to 1) [40]. This adversary model can be used in real-world linkage situations where the participating parties cannot be fully trusted, yet they cannot afford to be identified as a malicious party due to the proceeding consequences such as loss of reputation, facing legal charges, and financial loss.

2.5 Encoding and Encryption Techniques in PPRL

Different privacy techniques have been proposed for PPRL over the last few decades. These techniques can be divided into two main categories: (1) secure multi-party computation (SMC) based, and (2) perturbation based techniques. SMC based techniques enable several parties to be involved in a computation with their sensitive input values, where at the end of the computation no party learns anything about any other party's sensitive input values but all parties learn the final result of the computation [135]. While provably secure, SMC based techniques generally incur high computation and communication costs. Perturbation based techniques, on the other hand, transform and encode sensitive values in such ways that no sensitive information of entities can be obtained through those modified data [209]. Perturbation based techniques as used for PPRL generally have a trade-off between linkage quality, scalability to linking large databases, and privacy protection.

While some PPRL techniques focus on transforming or modifying sensitive data using one or more functions in such a way that the risk of reidentification of sensitive data is minimised, other techniques focus on establishing protocols that allow parties that participate in a linkage process to collaboratively and securely conduct a linkage. Different methods such as hash encoding, hash tables, bit representations, homomorphic encryption, and oblivious transfer, are being employed in different PPRL techniques [44, 177, 220]. Regardless of the methods used, the goal of any PPRL technique is to enable accurate and efficient linkage of records and at the same time preserve the privacy of the entities whose sensitive data are being linked.

Below we discuss these PPRL techniques in more detail. We mainly focus on perturbation based PPRL techniques since most of the existing privacy attacks exploit the vulnerabilities of such techniques rather than SMC based techniques.

2.5.1 Bloom Filter (BF) Encoding

Bloom filter (BF) encoding is a popular PPRL technique that uses hash functions to encode sensitive values and allows for approximate similarity calculations [177]. Even though BF encoding can be categorised as a hashing based technique (discussed in the following section), we discuss BF encoding here as a separate section in more detail because most of the existing work on PPRL are generally focused on BF encoding based methods. Further, in this thesis we employ BF encoding in multiple contribution chapters.

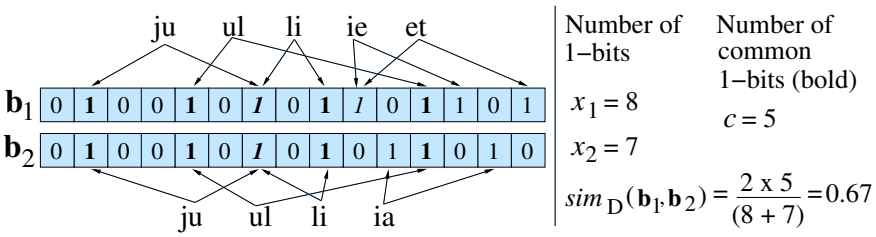


Figure 2.2: An example Dice coefficient similarity calculation between two BFs $\mathbf{b}_1$ and $\mathbf{b}_2$ which encode two *FirstName* values 'juliet' and 'julia', respectively. The strings are converted into bigrams ($l_q = 2$) and encoded into BFs of length $l = 15$ using $k = 2$ hash functions. The italic *1* at positions 7 and 10 in $\mathbf{b}_1$ and $\mathbf{b}_2$ are hash collisions.

A BF is a space efficient probabilistic data structure used to query set membership functions efficiently [24]. A bit vector of a specified length l is used as a BF where initially all bits in this vector are set to 0. The elements of a set are hashed into a BF using $k > 1$ independent hash functions, $\mathbf{h} = \{h_1, \ldots, h_k\}$. An element s in a set $\mathbf{s}$ is mapped into a BF $\mathbf{b}$ by setting the bit positions $\mathbf{b}[h_j(s)] = 1$, with $1 \leq j \leq k$, $1 \leq h_j(s) \leq l$, and $\forall s \in \mathbf{s}$. If a hash function returns a bit position that is already being set to 1 by any of the previous hash functions, we keep the bit position in $\mathbf{b}$ as 1. This type of events are called *BF collisions* [51].

The use of BF encoding in the context of PPRL was first proposed by Schnell et al. [177] in 2009. In PPRL, the quasi-identifying (QID) attribute values in databases are converted into a set of q-grams (substrings of length l_q). Each q-gram is then hashed into a BF using a set of $k \geq 1$ hash functions. To calculate the similarity between two BFs, a set based similarity measure, such as the Dice coefficient or Jaccard similarity [38], can be employed. An example of the BF encoding and the similarity calculation processes is shown in Figure 2.2. As illustrated, after encoding the two string values 'juliet' and 'julia' into the two BFs, $\mathbf{b}_1$ and $\mathbf{b}_2$, the similarity between these two BFs can be calculated with the Dice coefficient. Due to its ease of implementation and efficiency in calculating similarities, BF encoding is now being used in multiple practical PPRL applications [10, 29, 153].

Several methods of how to encode records into BFs have been proposed. Generating one BF per attribute, known as *attribute level* BF (ABF) [177], is one method which has the advantage of allowing one similarity to be calculated per attribute. ABFs are however vulnerable to frequency attacks as we discuss in Chapter 4 [42, 41]. Two encoding methods proposed to overcome this vulnerability are known as *cryptographic long term key* (CLK) [178] and *record level* BFs (RBF) [61]. In both of these methods several attributes are hashed into one BF using different techniques.

Double hashing (DH) and random hashing (RH) have been proposed for BF encoding to hash q-grams into BFs. In DH k different hash functions are simulated using only two independent hash functions [52]. This approach is however vulnerable to attacks as we discuss in Chapter 4 [147, 119]. To prevent against these attacks, random hashing (RH) can be used where the q-gram to be hashed into a BF is used as the seed of a pseudo random number generator used for hashing [180]. To improve

the privacy of BF encoding various hardening techniques such as salting [178], balancing [180], and XOR-folding [179] are also proposed. However, these techniques generally have a trade-off between improved resilience to attacks at the costs of linkage quality because similarities between BFs are likely distorted between hardened BFs. We extend the discussion of BF encoding to Appendix A where we describe different BF hashing, encoding, and hardening techniques in more detail.

2.5.2 Hashing based Techniques

In general these techniques utilise a one-way hash algorithm to convert plaintext values in a sensitive database into hash codes [155, 156]. The generated hash codes are assumed to be irreversible, because it is computationally expensive and practically difficult to identify the original QID values having access to only the hash codes. The message-digest (MD5) and secure hash algorithms (SHA1, SHA2, SHA3) are such one-way hash algorithms used widely in the context of cryptography [111]. However, due to the collision vulnerabilities identified in MD5 and SHA1, they are considered as cryptographically broken and unsuitable for further use [57, 128].

The first PPRL technique which used one-way hashing was proposed by Quantin et al. [155] where the authors have utilised two additional pads with a hash function such as SHA1 to generate irreversible hash codes for the QID values to prevent potential dictionary attacks [40]. The first pad is used by the DOs to hash the QID values in their own databases, and the resulting hash codes are then sent to a LU to conduct the linkage. Before conducting the linkage, the LU again hashes the hash codes it received from DOs using the second pad. Since this approach classifies record pairs using exact matching of hash values, phonetic encoding such as Soundex or NYSIIS [38] has been used to reduce the impact of phonetic variations. Furthermore, in order to securely transmit hash codes between the DOs and the LU, the authors have proposed to use a public key encryption algorithm such as RSA [137]. However, this approach is susceptible to frequency attacks since frequent QID values will result in frequent hash codes [40]

Churches and Christen [44] have proposed the first PPRL technique based on one-way hash functions that allows approximate matching of records. A QID value from a record is first converted into a set of q-grams (character substrings) of length l_q, which is then used to create the power set of this q-gram set. For instance, when $l_q = 2$ (bigrams) the q-gram set of the value 'smith' will be {sm, mi, it, th} and the power set will include all possible subsets of q-grams from this set. The created subsets of q-grams are then hashed using a one-way hash function such as SHA2. This results in a set of encodings for a single QID value. These sets of encodings, together with the lengths of the corresponding q-gram subsets, are then sent to a LU to calculate the Dice similarity [38] of encoded QID value pairs. Despite the excessive communication cost required because of the transmission of the hash codes generated using the power set of q-grams, this approach limits the frequency information available to a LU to conduct a privacy attack (acting as an inside adversary) because the LU does not receive the original QID values or their entire sets of q-grams [44].

A recently proposed PPRL technique by Randall et al. [163] creates multiple hash codes for each record using multiple combinations of QID values. Using the Fellegi and Sunter probabilistic weight calculation method [73], this approach first calculates a matching weight for each combination of QIDs in a database. For instance, if the database has a QID set $A^s = \{FirstName, LastName, StreetAddress\}$, there are $2^3 - 1 = 7$ possible (non-empty) combinations of QIDs. Based on the calculated matching weight for each combination of QIDs and a user defined weight threshold, a set of QID combinations (called match-keys) are selected. Next, for each record in the database the attribute values in each QID combination are concatenated into a single string and then hashed using a keyed cryptographic hash function such as keyed-hash message authentication code (HMAC) [115]. The linkage of records is then conducted using these hash values. Using a dataset of available ground truth links (true matches), a set of suitable match-keys can be selected to achieve high linkage quality [163]. Our work described in Chapter 6 shows how the generated encodings of this approach are vulnerable to a frequency based attack, and how such an attack can be prevented using few modifications to the original encoding approach.

Tabulation based min-hashing [186] is a recently proposed encoding technique aimed to be used as an alternative to BF encoding. The basic idea behind this method is to use tabulation based min-hashing to encode QID values into binary representations [127]. First l sets of look-up tables are created with each set containing k look-up tables, where l and k are user defined parameters. A key in a look-up table points to a random bit string. Each q-gram from a set of q-grams extracted from one or more QID values is then hashed using a one-way hash function. The hash code of a single q-gram is divided into k values which are used as keys into the set of look-up tables to obtain k random bit strings. All k random bit strings are then XORed to get a single bit for a q-gram. For all q-grams such a bit string is obtained, and the bit string with the minimum value is selected. This process is repeated using all l sets of look-up tables and finally the least significant bits of all obtained bit strings are concatenated to generate a final bit vector of length l [186]. These generated bit vectors are then used to calculate approximate similarities between records. The selection of only the least significant bits prevents mapping of frequent 1-bits to frequent q-grams resulting in improved privacy against frequency based cryptanalysis attacks, as we discuss in Chapter 4. However, tabulation min-hash encoding has a much higher computational complexity compared to BF encoding because of the creation of and traversing through a large number of look-up tables [158]. Furthermore, the selection of suitable values for the parameters l and k are not discussed in the original work.

To overcome the weaknesses in BF and tabulation min-hash based encodings, Ranbaduge et al. [158] have recently proposed an integer representation based encoding for PPRL using two hashing steps. Similar to BF and tabulation min-hash encoding, two-step hash encoding first extracts a set of q-grams from one or more QID values in a record. Then, in the first hashing step, using a user defined number of hash functions, a bit matrix is generated by hashing the set of q-grams into a bit vector using each hash function . In the second step of the encoding process, each column of the bit matrix which has at least one 1-bit is then hash mapped into a ran-

dom integer value. This results in one set of integers per record in a database, which are then used to calculate the similarities between records. Compared to BF and tabulation min-hash encoding, two-step hash encoding provides improved privacy protection while being computationally efficient [158].

A special type of encoding technique for linking records named statistical linkage key (SLK-581)[1] was proposed and used by the Australian Institute of Health and Welfare [110]. This technique employs four QID values, first name, last name, full date or birth, and gender, of a record to generate a hash value like code called a linkage key for that record. The statistical linkage key of a record is the concatenation of the second, third, and fifth letters of the family name, second and third letters of the first name, full date of birth as a string in the form of 'ddmmyyyy', and gender, '1' for male and '2' for female. For instance, for a record of a male 'Liam Brown' born in 4th February 1989, the statistical linkage key will be 'ronia040219891'. If the names are too short to extract all the required letters '2' will be used as a substitute for the missing letters. The character '9' is used for other missing values so that the length of the linkage key is always 14 characters.

2.5.3 Reference Value based Techniques

In reference value based PPRL, a set of values, either randomly generated or extracted from a publicly available database (such as a telephone directory or a public voter registration list), is used to calculate the similarities between that set of values and the QID values in a sensitive database. The calculated similarities are then analysed to identify matching record pairs across databases. Alternatively, reference values are also used in hardening techniques for BF encoding, as we discuss below.

Pang et al. [151] proposed a three-party PPRL protocol where records are linked based on a set of reference values The two DOs first calculate the similarities (or distances) between their own sensitive values and a set of reference values shared between the DOs, and then send the calculated similarities to a LU. Using the triangular inequality [1] to analyse the similarities, the LU can identify the matching record pairs.

A reference values based k-anonymous three-party blocking approach for PPRL was proposed by Karakasidis and Verykios [104]. A nearest neighbour clustering approach is employed to create clusters where values from a publicly available reference table, known to all the DOs, are clustered in such a way that each cluster consists of at least k similar reference values to ensure k-anonymous privacy. The DOs then map the sensitive QID values in their own databases to these clusters based on the similarities between QID values and reference values in the clusters. These similarities are calculated using the q-gram based Dice coefficient [38]. The generated clusters are then sent to a trusted LU to conduct the matching of record pairs across blocks.

Vatsalan and Christen [201] proposed a two-party PPRL protocol to securely identify matching record pairs using a public list of reference values. First the DOs calculate the similarities between reference values and their own QID values, and then

[1] https://meteor.aihw.gov.au/content/index.phtml/itemId/349510

bin these similarities in order to hide the actual similarity values. Bins are created between the values 0.0 and 1.0 and the number of bins can be selected by the DOs. Binning is necessary because exposing the original similarities to the DOs might lead to the reidentification of actual QID values. Using the reverse triangular inequality property [201], the DOs can then quantify the similarities between actual QID values without sharing them among themselves.

Vaiwsri et al. [199] recently proposed a reference value based hardening technique for BF encoding. A set of reference values is first extracted from a publicly available global database in such a way that no pair of selected reference values has a similarity higher than a user defined threshold. These reference values are then used in a BF hardening method called BLoom-and-flIP (BLIP) based randomisation [7], where bits in a BF are flipped based on the selected list of reference values for that BF. Using reference values for the BLIP based hardening ensures that the linkage quality of the hardened BFs does not decrease while obtaining improved privacy compared to standard BFs.

2.5.4 Embedding based Techniques

In embedding, QID values are mapped into a multidimensional vector space ensuring that the distances (or similarities) between the QID values are preserved and can be calculated using the vectors in that vector space.

Scannapieco et al. [173] proposed an embedding based three-party PPRL protocol for approximate matching of sensitive values. In the first step of the protocol the two DOs generate a set of random strings based on an agreed set of parameters, including the string length and the number of strings. These random strings are then used to generate a list of reference embeddings using a distance matrix such as Levenshtein edit distance [40, 129]. In the second step each DO embeds the QID values from its own records into a metric space using the generated list of embeddings and an embedding method, such as SparseMap [95]. In the last step of this approach, the embedded vectors from both DOs are sent to the LU which then calculates the distance between the embedded vectors and classify corresponding record pairs into matches and non-matches based on the calculated distances.

This protocol was extended into a two-party protocol by Yakout et al. [228] to achieve better performance and privacy. In this protocol each embedded vector of a sensitive value is first converted into a complex number, and these complex numbers are then exchanged between the DOs which allows the DOs to identify matching record pairs between their sensitive databases.

Bonomi et al. [26] have proposed a novel embedding technique based on frequent variable length substrings to map a string value into a real-valued vector space. In contrast to the random selection of strings in the approach by Scannapieco et al. [173], this embedding method employs a privacy preserving frequent pattern-mining approach to identify frequent variable length substrings, which are then used as the base for generating embeddings. Each DO involved in the protocol first generates its own base of substrings using the selected frequent pattern-mining algorithm. To

guarantee differential privacy [66], the counts of substrings are perturbed by Laplace noise [66] during the mining process. As a result, the DOs involved in the protocol can securely share their own bases to determine a common base. Once a common base is generated each DO embeds the QID values from its own sensitive database into a vector representation and sends the generated vectors to a LU to conduct the linkage of encoded records.

2.5.5 Differential Privacy based Techniques

Differential privacy is a mathematical definition that provides strong privacy guarantees for individuals whose records are being stored in a sensitive database [66, 68]. It ensures that no information about an individual is compromised while allowing accurate data analysis about a population. Differential privacy is usually achieved by introducing controlled randomness into the results of queries on sensitive data with a given probability. Methods such as Laplace distribution [66] and randomised response [218] are used to add noise to data in a probabilistic way. Recently differential privacy has been employed in PPRL techniques to ensure the privacy of entities.

He et al. [90] have proposed a privacy model named *output constrained differential privacy* that satisfies end-to-end privacy guarantee while allowing the truthful release of the output of a function. Based on this privacy model the authors have also proposed two protocols for PPRL, a local differential privacy protocol (LDPP) and a Laplace protocol (LP). In LDPP a blocking strategy that satisfies local differential privacy is used where each record in a sensitive database is hashed independently into a predefined set of bins. This can be achieved by using algorithms such as randomised response [68] or Johnson-Lindenstrauss transformation [32]. However, the authors argued that LDPP is not able to achieve the expected subquadratic efficiency in blocking and therefore proposed the LP for PPRL. In this protocol the DOs first agree upon a blocking function with a predefined number of blocks, and assign their own sensitive records into these blocks. A carefully chosen set of dummy records is then added into these blocks to ensure that the block sizes are differentially private. The added dummy records are chosen following Laplace distribution [66]. Finally, the DOs employ a secure two-party matching protocol [33] to classify the compared candidate record pairs into matches [90]. The authors have shown that the LP is able to achieve high recall, provable privacy, and subquadratic efficiency in terms of blocking records in a database compared to the LDPP.

The differential privacy model proposed by He et al. [90] was extended into a three-party setting by Rao et al. [166], where a third party is employed to conduct the matching of records between two sensitive databases. Similar to the LP by He et al. [90], the two DOs first block their own sensitive databases and add dummy records into blocks in order to hide non-matching records within blocks. Each DO then encrypts all records inside its blocks using a public key encryption technique [65] and sends the encrypted values to the third party. The DOs also create a differentially private synopses for each generated block which they send to the third party so that it can prune unnecessary comparisons between records. The third

party also randomises the encrypted values of each potentially matching record pair in such a way that it does not change the distances between records while achieving differential privacy. The third party then sends these record pairs to the DOs and notifies them about the number of secure comparisons required. Finally, a secure cryptographic matching protocol [166] is used by the DOs to collaboratively classify record pairs into matches.

2.5.6 Numerical Encoding Techniques

Many techniques have been proposed in PPRL to encode string values (as we discussed above), while limited work has been carried out to encode and compare numerical values. This is because techniques like BF encoding cannot be directly employed to encode numerical values such as age due to their different characteristics compared to textual values. For instance, the numerical values 99 and 100 do not contain any common digits even though the numerical difference between the two values is just 1.

Vatsalan and Christen [205] proposed a BF based encoding technique for numerical values. Instead of converting a numerical value into a set of q-grams, a set of numerical values that are closer to the actual numerical value to be encoded are used. This set of numerical values are then hashed into a BF using k hash functions. The set of numerical values to be used for a given number is decided based on two parameters d_{max} and d_{int}, where d_{max} is the maximum allowed difference between numerical values and d_{int} is the interval between numerical values in the set. For instance, for a pair of numbers 57 and 58, assuming $d_{max} = 4$ and $d_{int} = 1$, the generated two sets of numerical values will be {55, 56, **57**, 58, 59} and {56, 57, **58**, 59, 60}, respectively. These two sets are then hashed into two BFs using k hash functions. As argued by the authors, when the difference between pairs of numbers increases the overlap between two corresponding sets of numerical values will decrease, making two numerical values with a larger difference to have a low similarity. It is worth noting that the parameters d_{max} and d_{int} need to be set separately for different domains such as integer, floating point, and modulus data.

A bit vector based distance-aware encoding technique for numerical values was proposed by Karapiperis et al. [106, 107]. In their approach, first a threshold d_{max} is defined as the maximum allowed Euclidean distance between two numerical values. Then a list R of independent and identically distributed real numbers is randomly generated between the interval $[b_{min}, b_{max}]$, where $[b_{min}, b_{max}]$ is the minimum and maximum of the range of the expected real numbers to be encoded, respectively. This interval is decided based on the application domain of numbers that are to be encoded, such as age, height, or BMI. Each value $r_i \in R$ is then used to define a set of intervals T where an interval $t_i \in T$ is calculated as $t_i = [r_i - d_{max}, r_i + d_{max}]$. These intervals are then used as the hash functions to generate a bit vector of length $|T|$. If the numerical value to be encoded falls into a given interval then the corresponding bit is set to 1, and to 0 otherwise. These bit representations are finally used to calculate the similarities between pairs of numerical values [106, 107] similar to BFs.

2.5.7 Secure Multi-party Computation based Techniques

Secure multi-party computation (SMC) aims to securely perform computation and communication between different parties in a protocol [135]. SMC based PPRL techniques allow different parties involved in a linkage protocol to collaboratively link records without revealing their sensitive data [5, 157, 159]. At the end of the linkage process each participating party in the protocol will only learn the final result (the linked pairs of record identifiers), and no party is capable of inferring any additional information about any other party's sensitive data. The concept of performing a secure computation between two parties was first introduced by Yao [229] and this work was later extended by Goldreich [80] for more than two parties.

In the context of PPRL, several SMC based techniques have been proposed. A hybrid approach for PPRL that employs both SMC and perturbation based techniques was developed by Inan et al. [96]. In the first step of this protocol, the DOs generalise the records in their own sensitive databases using k-anonymity [171, 191]. These generalised records are then encoded using a one-way hash function and sent to a LU who then classifies record pairs into non-matches and potential matches. All potential matches are further processed by the DOs using a SMC approach based on homomorphic or commutative encryption [79] to calculate the actual similarities between record pairs. A record pair is classified as a match if the corresponding similarity is at least a user defined threshold value.

A locality sensitive hashing based three-party PPRL protocol using homomorphic encryption [79] for matching records was proposed by Karapiperis and Verykios [109]. The DOs first use Bloom filter encoding [177] or an embedding method in the Euclidean space [173] for anonymising the QID values in their sensitive databases. Next, a locality sensitive hashing based method is used to divide the anonymised records into separate blocks in order to reduce the number of required comparisons between record pairs in the two databases [38]. DOs then separately hash the keys of their own generated blocks, and these hashed keys along with corresponding record identifiers (IDs) in each of those blocks are shared with a third party to forms record ID pairs using similar blocking keys. A homomorphic Paillier cryptosystem [150] is then used between the DOs to securely calculate the similarities of record pairs identified by the third party.

Kuzu et al. [123] proposed a three-party protocol for PPRL where a third party is employed to group QID values (such as first and last names) based on values from a publicly available database. The third party first shares a set of generated cluster centroids with the DOs, which then map the records in their own databases to these clusters locally based on the distance of the QID values in records to these cluster centroids. To ensure the privacy of records inserted into blocks, a differential privacy mechanism is used where blocks are perturbed by adding noise drawn from a Laplace distribution [66]. The Paillier cryptosystem [150] is then used to securely compute the similarities between records. First, the third party creates a public/private key pair and sends the public key to both the DOs. One of the DOs encrypts the records in its blocks using its public key and sends those encrypted

records to other DO who then performs a set of homomorphic operations to compute encrypted similarities between records. These encrypted similarities along with the corresponding pairs of record identifiers are again sent to the third party who decrypts the encrypted similarities using its own private key. Based on a decision rule, the third party then classifies record pairs into matches and non-matches, and shares the matching pairs of record identifiers with the DOs.

Wen and Dong [220] proposed two PPRL protocols for two-party linkage where one protocol supports exact matching of records while the other supports approximate matching. Both protocols assume a HBC adversary model as we discussed in Section 2.4. In the exact matching protocol, the first DO encodes its sensitive records into a BF while the other DO encodes its sensitive records into a garbled BF [56]. A garbled BF is a variation of a BF which uses a vector of λ-bit strings instead of a vector of bits, where λ is a security parameter. An element in a set is split into λ-bit shares using an XOR based secret sharing scheme, and then hashed into a garbled BF using k hash functions. The two DOs then use a secure set intersection protocol called oblivious Bloom intersection [220] to compute the intersection between BF of the first database and the garbled BF of the second database. At the end of the protocol both DOs learn which record pairs are matching between the two databases. In the approximate matching protocol each DO uses a min-hash locality sensitive hashing [127] approach to generate a hash table using the q-grams extracted from the QID values in records. Similar to the exact matching protocol, the hash table of one database is then encoded into a BF while the hash table of the other database is encoded into a garbled BF. The approximate matching of records in two BFs is then conducted again using oblivious Bloom intersection protocol [220].

It is important to note that while SMC based techniques are provably secure, in general they are computationally more expensive than perturbation based techniques. This is because the computations in SMC are usually conducted using expensive cryptographic functions [79, 150]. Furthermore, SMC techniques generally require a large number of communication steps between parties in a linkage protocol which will not be applicable in some real-world linkage scenarios.

2.6 PPRL Evaluation Metrics

A given PPRL technique can be evaluated under three dimensions: linkage quality, scalability, and privacy. Linkage quality and scalability measure the effectiveness and efficiency of the PPRL process while privacy measures the privacy protection provided by a PPRL technique. Below we briefly discuss different evaluation measures employed under these three dimensions focusing on the measures used this thesis to evaluate our proposed attack methods.

2.6.1 Linkage Quality Metrics

It is necessary to have ground truth record pairs in the form of true matches and true non-matches in order to measure the quality of the linkage process [38]. Assuming

we have ground truth data, first the quality of the results of private blocking step can be measured using two metrics: pairs completeness and pairs quality. Pairs completeness is the fraction of candidate true matching record pairs generated by the blocking approach compared to all true matching record pairs between the databases being linked. Pairs quality is the fraction of candidate true matching record pairs generated by the blocking approach compared to all candidate record pairs generated by that approach.

As we discussed in Section 2.3, we assume that the private classification step of the PPRL process results in two classes of record pairs, matches and non-matches. There can be four possible classification outcomes when the classified matches and non-matches are compared to the ground truth data:

- **True Positives (TP)**: The number of record pairs classified as matches that are also in true matches.
- **False Positives (FP)**: The number of record pairs classified as matches that are actually true non-matches.
- **True Negatives (TN)**: The number of record pairs classified as non-matches that are also true non-matches.
- **False Negatives (FN)**: The number of record pairs classified as non-matches that are actually true matches.

Using the TP, FP, TN, and FN values, the quality of the results of private classification step can be measured by employing two metrics among others: precision and recall [40, 209]. Note that in the experimental evaluation conducted in Chapter 7 we also use both precision and recall to measure the accuracy of the reidentification of q-grams [216].

1. **Precision**: The fraction of correctly classified true matching record pairs compared to all record pairs classified as matches [37, 38, 194], calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

2. **Recall**: The fraction of correctly classified true matching record pairs compared to all true matches in the ground truth data [37, 38, 194], calculated as:

$$Recall = \frac{TP}{TP + FP} \tag{2.3}$$

Apart from precision and recall, the F-measure is also a commonly used metric to measure the effectiveness of a binary classification technique. The F-measure is the harmonic mean of precision and recall [38, 40]. However, recent research has shown that the F-measure is not suitable for comparing RL classification algorithms as it can assign different weights to precision and recall based on the number of classified

matches [89]. The area under the precision-recall curve (AUC-PR) can be used as an alternative to F-measure to assess the quality of a classification algorithm [39].

2.6.2 Scalability Metrics

The scalability of a PPRL technique is an important aspect when linking large real-world databases. Efficiency of a private blocking technique can be measured using reduction ratio, which calculates the reduction in the number of record pairs to be compared generated by the blocking process against all possible record pairs that can be generated across two databases [39].

Runtime and memory consumption are two commonly used measures to assess the scalability of a PPRL process. Runtime measures the time required for each step in a PPRL protocol, including the time taken to execute both automated and manual processes. Memory consumption measures the total memory usage of the algorithms employed in a PPRL technique.

Apart from the above measures, one can also theoretically analyse the worst-case computational complexity of a PPRL technique using the big-O notation [11]. The complexity of an algorithm used in a PPRL technique can be either one or a combination of logarithmic, linear, log-linear, polynomial, exponential, and factorial complexities.

2.6.3 Privacy Metrics

The aim of a privacy technique in a PPRL protocol is to protect the privacy of the entities whose records are being linked. Therefore, evaluating the level of privacy protection provided by a PPRL technique is important. Standard information theory metrics such as information gain (IG) and relative information gain (RIG) have been used to assess the privacy of PPRL techniques in terms of reidentifying encoded QID values [64, 103, 105].

Information Gain and Relative Information Gain

Durham et al. [63] proposed to use mutual information (also referred to as information gain) to measure how much information an adversary can infer about the plaintext QID values in a sensitive database given she has access to encoded QID values from the same database. Here entropy, a measure of the uncertainty of data, is used to quantify the dependence between the plaintext and encoded databases.

Information gain (IG) between two given discrete random variables X and Y is calculated as:

$$IG(X|Y) = H(X) - H(X|Y), \tag{2.4}$$

where $H(X)$ is the entropy of the variable X and $H(X|Y)$ is the conditional entropy between X and Y. In the context of PPRL, X is the frequency distribution of plaintext values of a QID (for instance first name) in the sensitive database $\mathbf{D}^s$ and Y is the corresponding frequency distribution of encoded values of the same QID in the encoded database $\mathbf{D}^e$. A lower IG value indicates that it is more difficult for an

adversary to reidentify any original plaintext values from having access to only the encoded version of those values.

Karakasidis et al. [105] later proposed to use relative information gain (RIG) as an alternative to IG. The authors argued that the value of IG differs relative to the size of the sensitive database $\mathbf{D}^s$, and RIG can therefore be used to obtain a normalised value for IG relative to the entropy of X. RIG can be calculated as:

$$RIG(X|Y) = \frac{IG(X|Y)}{H(X)}. \tag{2.5}$$

It is important to note that in the above calculations it is assumed that the plaintext database $\mathbf{D}^p$ that an adversary has access to is the same as the sensitive database $\mathbf{D}^s$, i.e. $\mathbf{D}^p \equiv \mathbf{D}^s$.

Disclosure Risk

Vatsalan et al. [204] discussed two limitations of the IG and RIG measures proposed by Durham et al. [63] and Karakasidis et al. [105]: (1) it is assumed that the plaintext database is equivalent to the sensitive database ($\mathbf{D}^p \equiv \mathbf{D}^s$), and (2) both IG and RIG only provide a single value for a QID value distribution in $\mathbf{D}^e$ rather than a record-wise assessment of privacy. To address these limitations, Vatsalan et al. [204] then proposed a set of disclosure risk (DR) based metrics for PPRL.

The authors first proposed a measure named probability of suspicion (PS), which calculates the number of records in $\mathbf{D}^p$ that matches with a record in $\mathbf{D}^e$. Given a record $r \in \mathbf{D}^e$ matches with m records in $\mathbf{D}^p$, the probability of suspicion for that record, r_{ps}, can be calculated as $1/m$. Given $n = |\mathbf{D}^p|$, the PS value of r can then be normalised as:

$$r_{ps} = \frac{1/m - 1/n}{1 - 1/n} \tag{2.6}$$

The authors then defined five different disclosure risk (DR) measures for an encoded database $\mathbf{D}^e$ based on the normalised PS values for each record in $\mathbf{D}^e$.

1. **Maximum disclosure risk** (DR_{Max}): Corresponds to the maximum PS value that can be calculated for any record $r \in \mathbf{D}^e$.

$$DR_{Max}(\mathbf{D}^e) = max(r_{ps} : \forall r \in \mathbf{D}^e), \tag{2.7}$$

   where $max()$ outputs the maximum value of a given list of values.

2. **Marketer disclosure risk** (DR_{Mark}): Defines how many records in $\mathbf{D}^e$ has the PS value of 1.0 and therefore can be exactly reidentified. Given there are a number of records with PS value of 1.0, the $DR_{Mark}(\mathbf{D}^e)$ can be calculated as:

$$DR_{Mark}(\mathbf{D}^e) = \frac{a}{|\mathbf{D}^e|}. \tag{2.8}$$

3. **Mean disclosure risk** (DR_{Mean}): Calculates the average of all the PS values of records in $\mathbf{D}^e$.

$$DR_{Mean}(\mathbf{D}^e) = \frac{\sum_{\forall r \in \mathbf{D}^e} r_{ps}}{|\mathbf{D}^e|}. \tag{2.9}$$

4. **Median disclosure risk** (DR_{Med}): Corresponds to the median PS value of the records $r \in \mathbf{D}^e$. If there are two median values the average of those two values are calculated to get the DR_{Med}.

$$DR_{Med}(\mathbf{D}^e) = med(r_{ps} : r \in \mathbf{D}^e), \tag{2.10}$$

   where $med()$ outputs the median value of a given list of values.

5. **User acceptance mean risk** (DR_{UAM}): If the DOs accept that the encoded data are not at risk if there are more than a defined number of records in $\mathbf{D}^p$ which match with a single record in $\mathbf{D}^e$, then the risk of disclosure can be eliminated from those values, and the PS value for those records will set to 0.0. User acceptance mean risk can then be calculated by obtaining the average of the remaining PS values.

2.7 Summary

In this chapter we first briefly discussed the process of record linkage (RL) and then detailed the main steps of privacy-preserving record linkage (PPRL), including the encoding process of sensitive databases. We next described existing PPRL techniques under six different categories focusing mainly on perturbation based techniques. We also described linkage quality, scalability, and privacy metrics used for the evaluation of PPRL techniques. In the following chapter we present a novel taxonomy of attacks on PPRL techniques based on what we have learned from this background study.

Chapter 3

A Taxonomy of Attacks on Privacy-Preserving Record Linkage

In this chapter we propose a taxonomy for analysing privacy attacks on privacy-preserving record linkage (PPRL) techniques, where we categorise attacks across twelve dimensions. In Section 3.1 we first highlight the importance of a taxonomy of privacy attacks on PPRL, and in Section 3.2 we then detail our contributions. In Section 3.3 we provide an overview of a general attack on PPRL and describe its main steps. In Section 3.4 we outline our taxonomy of attacks on PPRL. In Sections 3.5 to 3.7 we then discuss three aspects of the proposed taxonomy in detail. Finally, in Section 3.8 we provide a summary of our findings.

3.1 Introduction

As we discussed in Chapter 2, a variety of PPRL techniques have been proposed over the last two decades to enable secure linkage of sensitive databases [40]. In Section 2.5, we divided these techniques into two categories: (1) secure multi-party computation (SMC) based, and (2) perturbation based techniques, and discussed their strengths and limitations. While SMC based techniques generally have high computation and communication costs, perturbation based techniques have a trade-off between linkage quality, scalability, and privacy. Despite their limitations, perturbation based techniques, such as Bloom filter (BF) encoding [177], have gained popularity to be used in real-world applications because they are easy to implement and can be used to link large databases efficiently. However, research has shown that BF encoding is vulnerable to different privacy attacks [42, 121, 139, 147]. To overcome the privacy weaknesses of BF encoding, alternative perturbation based encoding techniques have recently been developed [158, 163, 186]. Nevertheless, some of these techniques have also shown to be susceptible to certain privacy attacks [46, 214, 215].

Different privacy attacks have been proposed not only in the area of PPRL but also in the contexts of privacy-preserving data mining, statistical disclosure control, and secure data publishing [53, 69]. The common characteristic of these attacks is that they aim to reidentify encoded, encrypted, and/or anonymised values in a sensitive

Table 3.1: Specific notation used throughout this chapter. For common notation used throughout the thesis, refer to the list of notation at the beginning of the thesis.

$\mathbf{D}^r$	Reidentified database
A^s, A^p	QID attribute names in $\mathbf{D}^s$ and $\mathbf{D}^p$
$\mathbf{p}$	Set of parameter values used by the encoding function $encode()$
$\mathbf{p}'$	Set of encoding parameter values that an adversary has access to
$\mathbf{a}$	Set of algorithms that an adversary has access to
$\mathbf{m}$	Set of metadata that an adversary has access to
$\mathbf{e}$	Set of encoded QID values where $\mathbf{e} \in \mathbf{E}$
$\mathbf{q}$	Set of plaintext QID values where $\mathbf{q} \in \mathbf{Q}^p$
ϵ	An entity where $\epsilon \in \mathcal{E}^p$
s	Secret key used in the encoding process

database in order to disclose the identities of entities (individuals) represented by these values. Furthermore, real-world data breaches [47, 144] have highlighted that any technique aimed at preserving the privacy of sensitive data needs to be carefully analysed before being used in real-world applications. Given there are different attack methods that can be applied on PPRL techniques as we describe in Chapter 4, it is important to explore the characteristics of such attacks in order to understand their applicability in real-world situations.

3.2 Contribution

In this chapter we propose a novel taxonomy of attacks on PPRL techniques, where we categorise attacks using twelve dimensions. These dimensions are based on adversarial, technical, and practical aspects. Our proposed taxonomy can be used to analyse different characteristics of privacy attacks and will help data custodians to better understand the reasons behind potential privacy breaches, the level of accessibility to information that an adversary requires in order to be able to successfully conduct a privacy attack, and to determine the privacy guarantees of PPRL techniques with respect to existing attacks. In Table 3.3 we then analyse and categorise existing attacks on PPRL using our proposed taxonomy. Finally, we provide a set of technical and non-technical recommendations for PPRL based on the three aspects of our taxonomy.

In the following sections we discuss the proposed taxonomy in more detail where Table 3.1 provides the specific notation and terminology we use in this chapter.

3.3 An Outline of Attacks on PPRL

In this section we discuss the basic building blocks of an attack on PPRL. We describe the underlying assumptions that an attack makes in order to successfully reidentify encoded sensitive values, and then provide an overview of the main steps of an

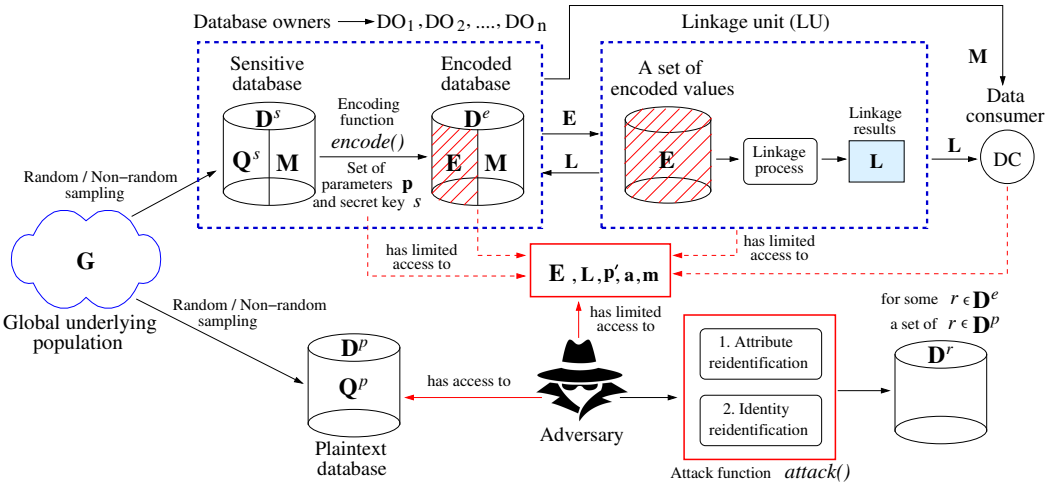


Figure 3.1: Overview of a privacy attack on PPRL. We use a single dotted line boxes to illustrate the steps of the database owners (DOs) (box on top left) and the linkage unit (LU) (box on top right) in the linkage protocol. The DOs first encode the quasi-identifying (QID) attributes, $\mathbf{Q}^s$, in their own sensitive database, $\mathbf{D}^s$, using an agreed encoding function $encode()$, a set of parameters $\mathbf{p}$, and a secret key s only known to the DOs. The encoded sets of QIDs, $\mathbf{E}$, from all DOs are then sent to a LU to conduct the linkage of these encoded QID values. After the linkage process the LU sends the identifiers of the matched records, $\mathbf{L}$, back to the DOs. We assume that an adversary obtains access to the encoded data, $\mathbf{E}$, along with some other information, such as parameter settings, $\mathbf{p}'$, a set of algorithms $\mathbf{a}$, and possibly metadata $\mathbf{m}$. We assume that the adversary also has access to a plaintext database $\mathbf{D}^p$ which she uses to perform the privacy attack on the set of encoded data $\mathbf{E}$. The adversary runs the attack function $attack()$, as we discuss in Section 3.3, with the aim to reidentify plaintext QID values for a set of encoded records in $\mathbf{E}$.

attack. The actual attack methods proposed on PPRL techniques following these steps are discussed in Chapter 4.

3.3.1 Underlying Assumptions

In Figure 3.1 we outline the fundamental steps of a privacy attack on PPRL. As illustrated, we assume a global underlying population $\mathbf{G}$ which is not seen by an adversary. We also assume a sensitive database $\mathbf{D}^s = (\mathbf{Q}^s, \mathbf{M})$ and a publicly available plaintext database $\mathbf{D}^p = \mathbf{Q}^p$, where we assume both have been sampled from $\mathbf{G}$. The sensitive database $\mathbf{D}^s$ can be owned by an organisation such as a hospital, a financial institution, a government department, or a security or military agency. $\mathbf{D}^s$ contains a list of quasi-identifying (QID) attribute values $\mathbf{Q}^s$ for a set of real-world entities $\mathcal{E}^s$. The sensitive database $\mathbf{D}^s$ also contains microdata of the entities $\mathcal{E}^s$, denoted by $\mathbf{M}$. These microdata can be any type of (sensitive) data containing medical, political, criminal, or financial information. Microdata are generally used in various data analytics domains such as data mining, decision making, fraud detection, and

health studies. We assume that the sensitive database $\mathbf{D}^s$ is deduplicated such that only one record representing a real-world entity is retained. The list $\mathbf{Q}^s$ is encoded using the encoding function $encode()$ which results in the encoded list of values $\mathbf{E}$, as we discussed in Section 2.3.

Similar to the sensitive database $\mathbf{D}^s$, the plaintext database $\mathbf{D}^p$ also contains a list of QID values $\mathbf{Q}^p$ for a set of real-world entities $\mathcal{E}^p$. We assume that an adversary has access to this plaintext database $\mathbf{D}^p$. Such a plaintext database can be sourced externally (such as a public telephone directory or voter database) or internally (if the attack is conducted by an insider, as we discuss in Section 3.5, who has access to a plaintext database that has some overlap with the sensitive database $\mathbf{D}^s$). The plaintext database $\mathbf{D}^p$ may or may not contain microdata. If $\mathbf{D}^p$ contains any microdata we assume that those are anonymised and cannot be used in the attack [53, 71].

3.3.2 The Main Steps of an Attack on PPRL

We now describe the main steps of a privacy attack on PPRL, as illustrated in Figure 3.1. Note that this description is based on the functionality of existing privacy attacks on PPRL. We assume that an adversary has access to the plaintext database $\mathbf{D}^p = \mathbf{Q}^p$ (as we discussed in the previous section) and the encoded database $\mathbf{D}^e = (\mathbf{E}, \mathbf{M})$. The adversary does not have access to the sensitive database $\mathbf{D}^s = (\mathbf{Q}^s, \mathbf{M})$, and therefore she does not know which set of encoded QID values $\mathbf{e}_i \in \mathbf{E}$ belongs to which set of plaintext QID values $\mathbf{q}_i \in \mathbf{Q}^p$, and which record $r \in \mathbf{D}^e$ represents which entity $\epsilon \in \mathcal{E}^p$.

We define an attack function, $attack()$, which consists of two steps. The first step is *attribute reidentification*, where the adversary runs a set of algorithms on the encoded data and aims to identify which plaintext QID values are encoded in each encoding $e \in \mathbf{E}$. For instance, in BF encoding [177], frequent bit patterns in a set of BFs from an encoded database can be used to identify frequent q-grams that have been encoded in each of those BFs [121]. As we discuss in Chapter 4, privacy attacks aim to identify a possible relationship between encoded and plaintext values in $\mathbf{D}^e$ and $\mathbf{D}^p$, respectively, and then use that relationship to reidentify encoded QID values.

The second step of the attack function $attack()$ is *identity reidentification*. Based on the QID values identified in the first step, a set of encodings in $\mathbf{E}$ is assigned to a set of records in $\mathbf{D}^p$. Essentially identity reidentification focuses on assigning encodings $e \in \mathbf{E}$ to real-world entities $\epsilon \in \mathcal{E}^p$. We can formally define the attack function $attack()$ as follows:

$$\mathbf{D}^r = attack(\mathbf{E}, \mathbf{D}^p, \mathbf{L}, \mathbf{p}', \mathbf{a}, \mathbf{m}), \tag{3.1}$$

where $\mathbf{L}$ is the linkage results, $\mathbf{p}'$ is a set of parameter settings, $\mathbf{a}$ is a set of encoding and attack algorithms, and $\mathbf{m}$ is the metadata of encoded sensitive data, that the adversary has access to. It is important to note that an adversary might not have access to all the information in Equation 3.1 in a given attack scenario. Different attacks have been proposed which require either all or some of this information.

The set of algorithms **a** can include encoding, hashing, frequency alignment, pattern recognition, or graph similarity matching algorithms [40], whereas the metadata **m** can contain information about data quality such as completeness and validity of the data, the date when the data was recorded, the database size, ownership details of the data, an so on. $\mathbf{D}^r$ is the reidentified database of records which is a subset of records $\mathbf{D}^e$. For each encoded record $r \in \mathbf{D}^r$, one or more plaintext records $r \in \mathbf{D}^p$ are potentially assigned by the attack.

In the following we propose a taxonomy of attacks on PPRL, which will help data custodians to, obtain a better understanding of existing privacy attacks, identify the limitations and weaknesses of attack methods, and, identify possible future research directions towards privacy attacks, as well as more secure encoding and encryption techniques for PPRL.

3.4 Taxonomy Overview of Attacks on PPRL

In this section we provide an overview of the taxonomy. As illustrated in Figure 3.2 there are three main aspects that can be considered to characterise an attack on PPRL, where we identify twelve dimensions under these three aspects.

- **Adversarial Aspects** focus on characterising the adversary, assumptions about the prior knowledge of the adversary, and non-technical issues related to the adversary. In Section 3.5.1 we describe three different adversary types depending on the information that an adversary has access to, and the collusion between different parties. The assumptions about an adversary's knowledge can be divided into two main dimensions: knowledge about the encoding parameters and assumptions about the database being attacked, as we discuss in Sections 3.5.2 and 3.5.3, respectively. In Section 3.5.4, we also discuss four non-technical issues that the adversary can utilise when obtaining information about encoding parameters and encoded data.

- **Technical Aspects** cover the dimensions attack type, attack scope, vulnerability, and complexity. In Section 3.6.1 we discuss different attack types which categorise an attack on PPRL in terms of the actual techniques used in an attack to reidentify encoded sensitive values, while in Section 3.6.2 we describe the attack scope which defines the actual encoding techniques that an attack can be applied upon. In Section 3.6.3 we then discuss different vulnerabilities of encoded values that are being exploited by the attack to reidentify encoded sensitive values. Finally, in Section 3.6.4 we describe how the complexity of an attack can be assessed using big-*O* notation.

- **Practical Aspects** are important in understanding how a privacy attack could be applied in a real-world situation. In Section 3.7.1 we discuss how the success of an attack can be quantified based on the numbers of correct attribute and identity reidentifications resulting from the attack, and in Section 3.7.2 we

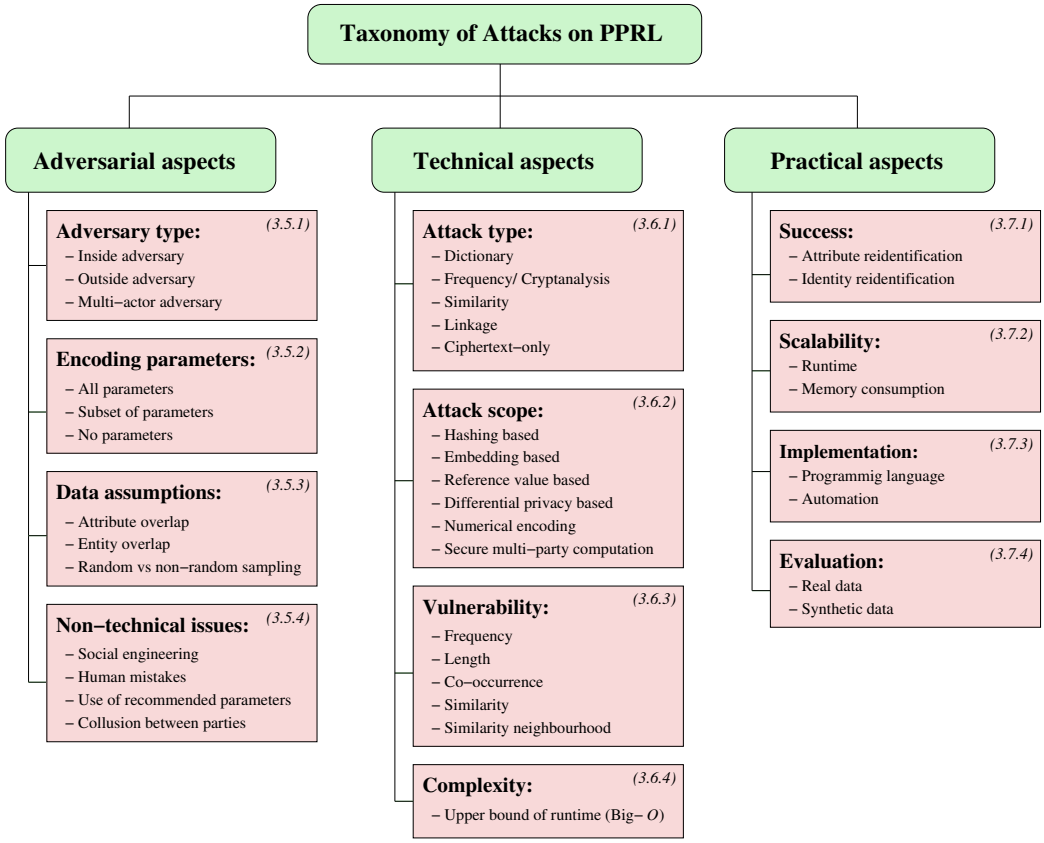


Figure 3.2: A taxonomy of attacks on PPRL described under three main aspects, where twelve dimensions are considered when characterising an attack. For each dimension we show the corresponding section number where we discuss a dimension.

discuss how the scalability of an attack can be measured using its runtime and memory consumption. In Sections 3.7.3 and 3.7.4 we then discuss different implementation techniques used to develop, and the databases used to evaluate existing attacks on PPRL, respectively.

In the following three sections we now discuss these twelve dimensions and describe their different aspects in more detail.

3.5 Adversarial Aspects

Here we discuss the different types of adversaries for an attack on PPRL, an adversary's accessibility to encoding function and parameter settings, data and domain related information, and possible impacts of non-technical issues for an attack.

3.5.1 Different Types of Adversaries

Based on the parties who are internal or external to the linkage protocol, as we discussed in Section 2.4, there can be three types of adversaries involved in an attack.

Inside Adversary (IA): An IA can be a person inside an organisation or a participant in a linkage protocol who tries to learn sensitive information about the entities whose records are being encoded. An IA might have access to the encoded database, linkage results, encoding algorithms, and potentially the encoding parameters and even the secret keys used [139]. All parties discussed in Section 2.4, namely database owner (DO), linkage unit (LU), and data consumer (DC), can be an IA. A DO itself can be an IA who aims to learn sensitive information about the encoded data of the other DOs. Since the LU has access to the encoded QIDs from different databases, it can conduct an attack trying to reidentify encoded values using a plaintext database. Depending on the linkage protocol, the LU might also have access to other information of the encoding such as parameter settings, algorithms, and metadata, which can be used in an attack [184].

A data consumer (DC) can also be a potential IA. For instance, a DC can be an employee who is given access to the microdata **M** and/or the linkage results **L** (such as the identifiers of record pairs that were classified as matches) to conduct different data analyses. A DC can, for instance, perform a similarity attack on **M** to reidentify unique entities by aligning values in **M** (such as zipcode and year of birth) with plaintext QID values based on the uniqueness of these values and their combinations [53]. An IA can also be someone who is not among the main parties in a linkage protocol. For instance, an ex-employee who used to work in linkage projects will potentially know details about previously used encoding functions and parameter settings. Furthermore, an employee inside the organisation might gain access to encoded values using company databases or another employee's login credentials [40].

Outside Adversary (OA): An external party who does not participate in a linkage protocol and does not have access to the encoding algorithms and/or encoding parameters used, yet who aims to learn about the sensitive information about the entities whose records are being linked, can be considered as an OA. An OA can gain access to encoded data, **E**, or microdata, **M**, in different ways. For instance, if **M** is made public after the encoding of QID values (under the assumption that no information could be linked to unique entities), that microdata can be accessed by possibly anyone. Since microdata can contain aggregated QID values such as age groups, gender, and/or zipcode, an OA can potentially conduct a linkage attack using another known plaintext database with similar QID values to reidentify entities [190], as we discuss in Section 3.6.

Alternatively, an OA can gain access to the encoded data as well as encoding parameter values illegally. For instance, data can be stolen from hard drives or flash drives, or an adversary can conduct a social engineering attack (as we discuss in Section 3.5.4) to gain access to encoding parameter settings, **p**, and even the secret key, s. An OA can potentially also eavesdrop on the communication channels between the

parties in a linkage protocol with the intention of learning secret information and/or interrupting such communication. It is important to note that if an OA gains access to the same encoding information that is accessible to an IA, then that OA can be seen as an IA.

Multi-actor Adversary (MA): A MA is a subset of participating parties in a linkage process (as discussed in Section 2.4) who can collude with each other to learn information about another party's encoded QID values. In such a collusion, each party contributes different pieces of information for an attack. This information can include details about the encoding process (such as encoding parameter settings, algorithms, and secret keys used), information about the sensitive databases, technical expertise, and sometimes even financial support [40]. There can be different types of collusion scenarios depending on the number of parties involved in the linkage process [162]. For instance, in a multi-party linkage protocol where there are more than two DOs, some of the DOs can collude with each other with the intention of learning information about another DO's sensitive values. Possible collusion of two parties of IAs include DO with DO, DO with LU, DO with DC, and LU with DC [162]. We discuss collusion in more detail in Section 3.5.4. In a MA attack the more parties collude with each other the more sensitive and confidential information could be compromised.

An IA can also seek additional information from a party outside a linkage protocol or the organisation, such as how certain algorithms work and how to attack encoded data, and/or aim to gain access to plaintext databases. However, we assume an outside party cannot provide any additional knowledge about the encoding process that an inside party does not already know.

3.5.2 Knowledge About Encoding Techniques and Parameters

Knowledge about the encoding includes identifying the actual encoding technique and the encoding parameters **p** used. Based on the encoded values in **E** the adversary might be able to guess which encoding technique was employed. For instance, recent PPRL encoding techniques such as two-step hash encoding [158] generate sets of integers for each record in a database, while the multiple match-key encoding technique [163] generates a set of hash values per record, obtained by hashing subsets of QID values. Since there are currently no other encoding techniques which generate similar outputs (to the best of our knowledge), the adversary can guess the encoding technique used by a simple examination of encoded values. On the other hand, both BF encoding [177] and tabulation min-hash encoding [186] result in a list of bit vectors as encodings of QID values. Therefore, it will be difficult for an adversary to guess which encoding technique (BF or tabulation min-hash encoding) was used if she only has access to an encoded database of bit vectors.

The encoding parameters **p** can contain of different settings depending on the actual encoding technique used. For instance, parameters used for BF encoding are the BF length, q-gram length, number of hash functions, the hashing method, and the encoding technique [177]. Some of these parameters, such as BF length, can be obtained by an adversary using the length of the encoded values. The number of

hash functions used can also be learned in certain situations by analysing the Hamming weight of the encoded values. However, it is difficult to guess other parameter settings even using an analysis of encoded values.

Given $\mathbf{p}$ and s, the set of parameter values and the secret key used for the encoding, respectively, and $\mathbf{p}'$, the set of parameter values that the adversary has access to (or can guess), we can define three scenarios:

1. $\mathbf{p}' \equiv \mathbf{p}$ and s: The adversary knows all the encoding parameters including any secret keys used in the encoding process. It is important to note that for certain encoding methods (such as BF encoding), having access to all the parameter settings does not necessarily mean that the adversary can simply reverse-engineer the encoding process to reidentify all values in an encoded database. An example for such an attack was proposed by Mitchell et al. [139] on BF encoding where it was assumed that $\mathbf{p}' \equiv \mathbf{p}$ and s.

2. $\mathbf{p}' \subset \mathbf{p}$: The adversary knows a subset of the encoding parameters. This does not include any secret keys used. Most of the existing attacks on PPRL are proposed under this scenario. Different attacks make different assumptions about an adversary's actual knowledge of $\mathbf{p}$, as we discuss further in Chapter 4. Such prior knowledge on parameter settings allows the adversary to efficiently conduct a privacy attack without having to do an exhaustive search on all possible parameter settings.

3. $\mathbf{p}' = \emptyset$: The adversary does not know any of the encoding parameters. Without having access to at least some of the parameter settings used for the encoding, most existing attacks on PPRL will be impossible to perform. In such a scenario the adversary might need to consider all possible combinations of values for different parameter settings, therefore making the attack not practical. However, even without having access to any of the parameter settings used, with certain encodings, the adversary can potentially guess the encoding function, $encode()$, as we described above. Furthermore, real-world incidents based on linkage attacks [47, 144, 190], as we discuss in Section 4.3, have shown that techniques like generalisation and randomisation can be susceptible to privacy attacks even without the knowledge of the parameter settings, $\mathbf{p}$, used.

3.5.3 Availability and Accessibility to Data

All existing privacy attacks on PPRL assume that the adversary has access to a plaintext database $\mathbf{D}^p$ which has some degree of overlap with the sensitive database $\mathbf{D}^s$, and therefore by extension with the encoded database $\mathbf{D}^e$. With $\mathbf{D}^p$ the adversary can aim to reidentify values in the encoded database by performing, for example, frequency alignments between plaintext and encoded values [41, 216]. However, depending upon the actual overlap of the two databases $\mathbf{D}^p$ and $\mathbf{D}^s$, the difficulty of conducting an attack can vary.

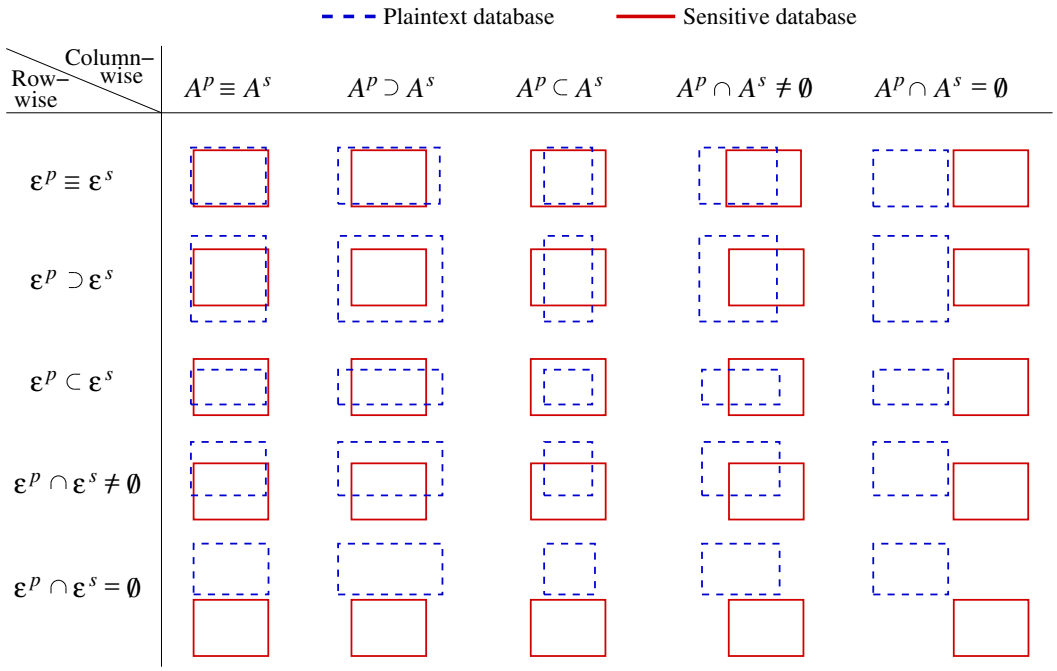


Figure 3.3: All 25 combinations of column-wise and row-wise overlap between the sensitive and plaintext databases $\mathbf{D}^s$ and $\mathbf{D}^p$. We consider the overlaps between the two sets of QID names A^s and A^p, and the two sets of entities $\mathcal{E}^s$ and $\mathcal{E}^p$ in column-wise and row-wise overlap, respectively. Each combination illustrates a possible scenario of an adversary's accessibility to plaintext values.

Conceptually a database can be considered as a matrix with rows corresponds to the actual records representing individuals in the database while columns corresponds to the different attributes (including QIDs), where each individual record (row) has values for these attributes. A privacy attack can be more successful if the plaintext database $\mathbf{D}^p$ overlaps in both rows and columns with the sensitive database $\mathbf{D}^s$ because as the overlap between the records and the QIDs in these two databases increases, there is a higher possibility that more encoded values in $\mathbf{D}^e$ can be reidentified. In Figure 3.3, we show 25 different scenarios of an adversary's accessibility to a plaintext and an encoded database. Each scenario represent a different overlap of QIDs and records between the plaintext database $\mathbf{D}^p$ and the sensitive database $\mathbf{D}^s$.

Column-wise Overlap: The overlap between columns in $\mathbf{D}^p$ and $\mathbf{D}^s$ determines how much information about QIDs, such as *FirstName*, *LastName*, and *StreetAddress*, are similar in both databases. Assuming A^s and A^p are the sets of QIDs in $\mathbf{D}^s$ and $\mathbf{D}^p$, respectively, we define the five different scenarios of availability of QID information to an adversary as shown in the top row of Figure 3.3. The amount of information that the adversary has will reduce with less overlap between QID names. Therefore, when going from the first to the fifth scenario the difficulty of an attack will increase.

1. $A^p \equiv A^s$: The adversary has access to a set of QIDs from $\mathbf{D}^p$ which is equivalent to the set of QIDs in the sensitive database $\mathbf{D}^s$. Depending on the number of common records in the two databases, as we discuss below in row-wise overlap, in this scenario there will likely to be a high overlap between QID values across both databases.

2. $A^p \supset A^s$: The adversary has access to a set of QIDs from $\mathbf{D}^p$ which is a superset of the set of QIDs in $\mathbf{D}^s$. Compared to the above scenario, the adversary has access to the same set of QIDs used in $\mathbf{D}^s$ along with some additional QIDs. This scenario will be different from the above scenario when the adversary does not have specific knowledge about the QIDs used for the encoding.

3. $A^p \subset A^s$: The adversary has access to a set of QIDs from $\mathbf{D}^p$ which is a subset of the QIDs in $\mathbf{D}^s$. For instance, the set of QIDs in $\mathbf{D}^s$ can be $A^s = \{$*FirstName, LastName, StreetAddress*$\}$ while the set of QID in $\mathbf{D}^p$ can be $A^p = \{$*FirstName, LastName*$\}$. An analysis of frequency distributions of QID values in this scenario can potentially be of limited use since the adversary does not have all the required QID attributes in $\mathbf{D}^p$.

4. $A^p \cap A^s \neq \emptyset$: The adversary has access to a set of QIDs from $\mathbf{D}^p$ which has a non-empty intersection with the set of QIDs in $\mathbf{D}^s$. For instance, $\mathbf{D}^s$ has the set of QIDs $A^s = \{$*FirstName, LastName, StreetAddress*$\}$, whereas $\mathbf{D}^p$ has the set of QIDs $A^p = \{$*FirstName, LastName, Zipcode*$\}$. If the adversary does not know the QIDs used in the encoding process, conducting an attack will be difficult in this scenario compared to the previous scenarios.

5. $A^p \cap A^s = \emptyset$: The adversary has access to a set of QIDs from $\mathbf{D}^p$ which is not equivalent to the set of QIDs in $\mathbf{D}^s$. The availability of information that can be exploited in such a scenario, such as frequency information or relationships in similarities, will likely be less compared to above four scenarios. However, having access to QIDs that are closely related to QIDs in the sensitive database, for example *FirstName* $\in A^s$ and *MiddleName* $\in A^p$, can potentially lead to a frequency alignment of QID values or frequent q-grams.

Row-wise Overlap: The individual records represented by the rows in the databases allow an adversary to perform different operations such as frequency or similarity distribution analysis of QID values or q-grams. Such operations are usually the fundamental building blocks of a privacy attack [40]. If there is no overlap in the rows of the two databases $\mathbf{D}^s$ and $\mathbf{D}^p$ then conducting a privacy attack will be difficult. Here we assume that the rows in both databases $\mathbf{D}^s$ and $\mathbf{D}^p$ are recorded at the same point in time with perfect data quality and therefore there are no differences between the QID values of the same real-world entity. However, in practice there can be significance temporal differences between $\mathbf{D}^s$ and $\mathbf{D}^p$ (such as two census snapshots recorded with a large time gap between them), which can lead to differences in QID values such as *LastName* and *StreetAddress* for the same entity. Furthermore, there can also be various types of errors and missing data in the QID values of real-world

databases [40]. Assuming each row in the sensitive and plaintext databases are from two sets of real-world entities, $\mathcal{E}^s$ and $\mathcal{E}^p$, respectively, there can be five scenarios of accessibility as shown in the left most column of Figure 3.3. Similar to column-wise overlap, the less the overlap between entities (rows), the less the amount of information that the adversary will have access to and therefore the difficulty of an attack will likely increase.

1. $\mathcal{E}^p \equiv \mathcal{E}^s$: The entities (rows) in both $\mathbf{D}^s$ and $\mathbf{D}^p$ are equivalent. If these databases also have similar QIDs (the fist four scenarios as we discussed above for the column-wise overlaps) their frequency distributions will be highly similar in this scenario.

2. $\mathcal{E}^p \supset \mathcal{E}^s$: The entities recorded in $\mathbf{D}^p$ are a superset of the entities recorded in $\mathbf{D}^s$. Unlike the first scenario mentioned above, in this scenario the adversary has an extra set of entities in $\mathbf{D}^p$. The difficulty of conducting an attack in this scenario (and the next two scenarios) depends on two additional factors: (1) the size of the row-wise overlap (the number of common records) and (2) the sampling process of the two databases $\mathbf{D}^p$ and $\mathbf{D}^s$ (random vs non-random) as we discuss below. If the size of the row-wise overlap is larger than the number of non-overlapping records, and if the two databases are assumed to be sampled at random from the global population $\mathbf{G}$, then the frequency distributions of QID values will still be similar in both databases [178]. However, smaller row-wise overlap as well as a non-random sampling of two databases can have an impact on the frequency distributions of values and therefore can increase the difficulty of an attack.

3. $\mathcal{E}^p \subset \mathcal{E}^s$: In this scenario the entities in the plaintext database are a subset of the entities in the sensitive database. Similar to the above scenario, the similarity in frequency distributions of the two databases $\mathbf{D}^p$ and $\mathbf{D}^s$ depend on the size of the row-wise overlap and the sampling process of the two databases.

4. $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$: There is a non-empty overlap between records (rows) in the encoded and plaintext databases. However, both databases contain entities that are not common. Hence, this scenario can be considered as a more practical scenario related to many real-world situations. Even with non-common records in both databases, if the number of overlapping records is larger than the number of non-overlapping records, there will likely be similar QID value distributions [178] and the adversary can perform a frequency analysis of QID values to reidentify encoded QID values in $\mathbf{D}^e$.

5. $\mathcal{E}^p \cap \mathcal{E}^s = \emptyset$: The overlap between records in the plaintext and sensitive database is empty. Hence, there are no real-world entities common in the two databases. Compared to the previous scenarios, conducting an attack in this situation will likely be more difficult. Without enough frequency information of QID values, most existing privacy attacks cannot be performed [42, 121, 147]. However, if the two databases are considered as samples from a population with similar

QID value distributions across different categories (such as two samples from different age categories in the same state with similar name distributions), an attack can still likely be performed. For instance, if the sensitive database has two records with the name 'Peter Miller' and three records with the name 'Jackson Miller', and the plaintext database has five records with the name 'James Miller', even though the two databases do not contain records about the same entities the frequencies of the value 'Miller' will be the same.

Random vs non-random sampling: In Figure 3.1 we show that both the sensitive $\mathbf{D}^s$ and the plaintext $\mathbf{D}^p$ databases can be sampled from the underlying population $\mathbf{G}$ either randomly or non-randomly. While random sampling might be considered in a theoretical analysis, this is likely not the case in a practical real-world situation. For instance, assume $\mathbf{D}^s$ is a health database from a hospital in a specific state and $\mathbf{D}^p$ is a publicly available voter registration database from the same state. While the health database will include records about children, non-citizen residents, and travellers, the voter registration database will only contain records about adult citizens. These aspects will influence the randomness of the samples in both of these databases.

If both $\mathbf{D}^s$ and $\mathbf{D}^p$ are sampled randomly, every entity in $\mathbf{G}$ has a non-zero probability of being selected for $\mathbf{D}^s$ and $\mathbf{D}^p$. Therefore, the characteristics, such as frequency distribution of QID values, of these databases will likely be more similar and the adversary will have access to those characteristics which can be used in an attack. On the other hand if the databases are sampled non-randomly then different subsets of $\mathbf{G}$ will likely be selected for $\mathbf{D}^s$ and $\mathbf{D}^p$ and therefore the difficulty of an attack will increase because of potential biases that can be introduced resulting different characteristics in $\mathbf{D}^s$ and $\mathbf{D}^p$ such as different frequency distributions of QID values.

3.5.4 Non-Technical Adversary Issues

Non-technical issues focus on how human behaviour can affect both privacy and security[1] of a PPRL system. The disclosure of information via non-technical means can sometime bypass all technical security measures that have been implemented to protect the sensitive data [222]. Furthermore, certain non-technical issues we discuss below can lead to an OA to become an IA by obtaining confidential information about a PPRL protocol. Therefore, recognising such non-technical issues is as important as technical issues in order to control and minimise the privacy risks associated with a PPRL project. In the following we discuss four such non-technical issues that potentially can be employed by an adversary in a privacy attack.

Social Engineering: Social engineering is defined as the malicious manipulation of people through human interaction to gather confidential and/or harming information [86]. Techniques used in social engineering attacks include phishing, vishing, smishing, pretexting, and so on [230]. Phishing is a commonly used technique where

[1]While privacy is concerned with protecting the identities of individuals when using their personal information, security is focused on safeguarding the data as well as the systems that those data are stored in [16].

an adversary sends a fraudulent email or message that appears to come from a legitimate business or a government department asking for confidential information such as login credentials or credit card details. In vishing (voice phishing) the attack happens over a phone call, while in smishing (short message service phishing) the attack happens using a mobile phone text message. In pretexting the adversary impersonates a legitimate party or creates a fabricated scenario and tries to trick people into giving private information [15] such as bank details, answers to security questions, and trade secrets.

Almost all existing privacy attacks proposed on PPRL make certain assumptions about an adversary's prior knowledge about the used encoding techniques [42, 119, 139, 147]. Through social engineering attacks an adversary can potentially gain access to such information, as we discussed in Section 3.3.2.

Human Mistakes: Human mistakes (also referred to as human error) is a notable fact that should be taken into consideration in PPRL, yet it is often disregarded as trivial. Human mistakes refer to an action of a person that is not intended, not expected by a set of rules or an external observer, and that moves a task or a system outside its acceptable limits[2]. Human mistakes can occur due to bad decision making, stressful environments, human-computer interface issues, and poor use of available resources. An empirical study done by Liginlal et al. [134] showed that human mistakes in information processing constitute for most cases of privacy breaches.

In PPRL, human mistakes can occur in three main ways: (1) the use of highly predictable secret keys/passwords, (2) the use of PPRL techniques and parameter settings that have been shown to be vulnerable to privacy attacks, and (3) allowing an OA to become an IA due to social engineering. Despite the warnings not to, people tend to use passwords or secret keys that can be easily remembered, such as dates of births and names of partners or family members. An adversary can potentially predict or learn such passwords. In a PPRL technique that utilises a shared secret key, if the key is compromised, the adversary can perform a dictionary attack to potentially learn the remaining parameters used for the encoding of values [139].

The second type of human mistake is the use of PPRL techniques and parameter settings that are susceptible to privacy attacks. Although certain PPRL techniques, such as one-way hash encoding, can be successfully attacked [215], some organisations might be using these techniques. This might be due to the ease of implementation, or the organisation has built their infrastructure specific to that method. However, such decisions can make an attack easier.

The third type of human mistake is letting an OA become an IA. For instance, an ex-employee of an organisation can gain access to encoded values and/or encoding parameter settings just because the responsible authorities have failed to change the system passwords or remove the ex-employee from the list of authorised users. Furthermore, confidential information about the encoding parameters can be disclosed from flash drives or personal computers that are not password protected. Such incidents often occur due to irresponsible behaviour of people [134].

[2] https://en.wikipedia.org/wiki/Human_error

Use of Recommended Parameter Settings: The use of recommended, commonly known, or published parameter settings is another non-technical issue that can potentially help an adversary. Often in publications where PPRL techniques are proposed, a set of parameter settings are recommended. In a real-world application of that PPRL method, if the same set of parameter settings are employed, then an adversary can guess those parameter settings just by referring to the relevant publication. Even if the PPRL technique did not use the exact same set of recommended parameters, the published settings will potentially allow the adversary with a likely starting point to estimate the actual parameter values.

Collusion between Parties: In a collusion scenario, two or more parties in a linkage protocol (from multiple actors as we discussed in Section 3.5.1) work together to learn information about another party's sensitive data by sharing their own information such as the parameter settings used in the encoding process [162, 200]. As shown in Table 3.3, most existing privacy attacks on PPRL make different assumptions on an adversary's prior knowledge about certain parameter settings used in the encoding. One possibility of attaining encoding details is via collusion between parties. Collusion might happen for various reasons, which include financial gain, intellectual gain, and personal vendetta. For instance, a collusion between a DO and a researcher (DC) can be inspired by the additional information that the researcher can gain access to in order to conduct more sophisticated data analytics, while the DO can learn more about the matched records of another DO. Similarly, a collusion between a DO and a LU can be inspired by the secret encoding parameter settings that the LU can learn from the DO to potentially mount a dictionary attack on encoded data from other DOs, while a DO participating in a collusion can gain access to the encoded data from other DOs that participate in the protocol.

3.6 Technical Aspects

Here we describe the main technical aspects related to attacks on PPRL, including attack types and their scope, the different vulnerabilities exploited by these attacks, and measuring the complexity of an attack. We also discuss the possibility of linkage attacks and ciphertext-only attacks in a PPRL protocol.

3.6.1 Attack Types

Based on the various techniques used, privacy attacks on PPRL can be divided into three categories. Note that some existing attacks can be placed into more than one category because they use a combination of different techniques.

- **Dictionary Attacks**: In a dictionary attack an encoded QID value is reidentified by encoding a large number of likely possible plaintext values in a given list of values until an encoding that matches is found. In the context of PPRL a dictionary attack is possible when the adversary has access to the encoding function

encode() and the used set of parameters **p**. In such a scenario the adversary can use *encode*() and **p** to encode all QID values in a plaintext database and see which plaintext encoding matches with encoded values from the sensitive database. However, a dictionary attack becomes difficult if the adversary does not have access to **p**, or if the encoding technique uses a secret key s [147] (likely unknown to the adversary) to generate these encodings. In such a scenario a brute force attack should be conducted using all combinations of possible values for s and/or **p**.

- **Frequency Analysis based Attacks**: This type of attack computes the frequency distributions of values in **E** and compares those distributions with the distributions of values in $\mathbf{D}^p$. Frequency analysis of combinations of characters in different languages is commonly used to break classical ciphers such as substitution and Vigener ciphers [185]. A frequency analysis can be conducted without requiring any or using only limited knowledge about the encoding function *encode*() and its parameters **p**. As we discussed in Section 3.5.3, having similar frequency distributions in $\mathbf{D}^p$ and **E** will depend on the row-wise overlap and the random/non-random sampling of the databases. Furthermore, when several QID values in a record are encoded into a single encoding (such as record-level BFs [61]), then the alignment of frequent values will not be enough to accurately reidentify encoded values. This is because only a limited amount of frequency information will be available that can be extracted [119].

  A cryptanalysis attack is a variation of a frequency attack that utilises additional knowledge about an encoding technique and other methods apart from the basic frequency alignments to reidentify encoded values [40]. In general, a cryptanalysis attack starts with a basic frequency alignment of plaintext and encoded values. The results of such an alignment are then used by other methods to successfully reidentify encoded sensitive values. These methods can include constraint satisfaction models [121], language models [119], correlation analysis [41], and pattern-mining [216]. Most existing privacy attacks proposed on BF encoding for PPRL can be categorised as frequency based cryptanalysis attacks, as we describe further in Chapter 4.

- **Similarity Attacks**: Most PPRL encoding techniques calculate approximate similarities between encoded records as we discussed in Section 2.5. Approximate similarities can also be calculated for plaintext QID values (using corresponding sets of q-grams) in a plaintext database. Often these similarities calculated using the plaintext and the corresponding encoded values have an approximate linear relationship [214]. A similarity attack can exploit such relationships between similarity distributions to match an encoded value with a plaintext QID value. One way of exploiting such relationships is using similarity graphs to perform a matching of nodes across graphs [46, 214].

There are a number of other types of attacks that are not directly related to PPRL, but that have been used in the contexts of privacy-preserving data publishing and

cryptography [53, 111]. In the following we describe two such attack types and briefly discuss why they have not been used in the context of PPRL.

Linkage Attacks: This type of attack attempts to reidentify entities in an anonymised database by linking anonymised records with available external information (such as a public database) based on the uniqueness of the values in those records. Unlike the other attack types, a linkage attack is not conducted on encoded values in the PPRL context, rather it can be conducted on generalised and/or randomised QID values or microdata [53]. Such QID values might include zipcode, year of birth, gender, and age group. A DC can conduct a linkage attack, or a collusion between a DO and a DC can also facilitate a linkage attack as we discussed in Section 3.5.1, because a DC will potentially have access to the microdata $\mathbf{M}$ from all DOs. No linkage attacks have been proposed in the context of PPRL because PPRL techniques are focused on encoding sensitive information, not on generalising and publishing them. In Section 4.3 we describe three real-world linkage attacks proposed in the context of privacy-preserving data publishing.

Ciphertext-only Attacks: In cryptography, ciphertext-only attacks are a basic type of attack where an adversary attempts to recover the plaintext values and/or the secret key using only the ciphertext [111]. In such an attack, the adversary exploits common characteristics of a certain language such as redundancies in words and common occurrences of characters. This type of attack is different from known plaintext attacks because in a ciphertext-only attack the adversary does not have access to any plaintext data that she can use in the attack.

In the context of PPRL, to the best of our knowledge, no attack has been proposed so far to reidentify encoded values in $\mathbf{D}^e$ without using a plaintext database $\mathbf{D}^p$. This is because PPRL techniques encode QID values of people such as their names and addresses and the characteristics of such values can differ from population to population, and they can also change over time. Therefore, a successful attack cannot be conducted using only encoded data without having access to a plaintext database.

3.6.2 Attack Scope

The scope of an attack on PPRL defines what PPRL techniques the attack can be applied on. As we discussed in Section 3.1, PPRL techniques can be categorised into SMC based and perturbation based techniques. All existing attacks on PPRL have been proposed for perturbation based techniques because these techniques do not have provable privacy guarantees, and different privacy weaknesses exist in perturbation based techniques that can be exploited. However, if a SMC based technique calculates approximate similarities of records, then that technique can potentially be attacked using a similarity attack. As discussed in Section 2.5 we can divide perturbation based techniques into five categories:

- Hashing based techniques use one-way hash functions such as secure hash algorithms (SHA) [111] to either convert sensitive plaintext values into hash codes, or to map plaintext values into bit or numerical vectors [44, 65, 163, 177].

- Embedding based techniques map plaintext values into multi-dimensional vectors [173, 228], where these vectors are then used to calculate the similarities between records.
- Reference value based techniques use publicly available lists of reference values to calculate similarities between sensitive QID values and reference values [151, 200]. These similarities are then shared between the DOs or sent to a LU to calculate the similarities between records.
- Differential privacy based techniques focus on anonymising QID values in a sensitive database by blocking values such that each block satisfies certain differential privacy requirements [90, 166]. The generated blocks are anonymised by adding noise, most commonly from the Laplace distribution [66].
- Numerical encoding techniques focus on encoding numerical values of such a way that the distances between numerical values are preserved [106, 205].

Almost all existing privacy attacks on PPRL exploit the weaknesses in hashing based encoding techniques and their different parameter settings [42, 119, 121, 214, 215], as we describe in Chapter 4.

3.6.3 Vulnerability

In a privacy attack, an adversary can exploit one or more vulnerabilities of values in both the plaintext ($\mathbf{D}^p$) and the encoded ($\mathbf{D}^e$) databases in order to reidentify encoded sensitive values. A value in either $\mathbf{D}^p$ or $\mathbf{D}^e$ becomes vulnerable based on how unique or rare it is in its corresponding database. Following the concept of k-anonymity [171, 172], five types of vulnerabilities can be defined: (1) frequency vulnerability, (2) length vulnerability, (3) similarity neighbourhood vulnerability, (1) co-occurrence vulnerability, and (2) similarity vulnerability. We discuss these vulnerabilities in detail in Chapter 5 where we propose a vulnerability assessment framework for PPRL.

3.6.4 Complexity

A theoretical analysis of computational complexity of an attack can be performed using the big-O notation [114]. The big-O notation describes the asymptotic upper bound for the complexity requirements of an algorithm in terms of the size of the input. Assuming $n = |\mathbf{D}^e|$ records in an encoded database, the big-O notations of $O(log\ n)$, $O(n)$, $O(n\ log\ n)$, $O(n^c)$, $O(c^n)$, and $O(n!)$ represents logarithmic, linear, log-linear, polynomial, exponential, and factorial complexities, respectively. Only some attacks in the literature have been analysed with regard to their computational complexity in terms of the big-O notation.

It is worth noting that one could view the computational complexity as an attribute of the adversary, that is, the adversary's computational power. However, here we specifically discuss about the technical aspects related to an attack itself. Further,

under practical aspects we discuss about the scalability of an attack (in Section 3.7.2) which can be attributed to the computational power that the adversary has access to.

3.7 Practical Aspects

In this section we describe how an attack can be evaluated based on its reidentification accuracy and how scalable an attack is when applied on (large) real-world databases. We also discuss the implementation aspects of an attack and the databases that have been used to evaluate attacks on PPRL.

3.7.1 Success of an Attack

Assessing the success of a privacy attack can be used to quantify the privacy protection provided by PPRL encoding technique the attack has been applied on. However, neither a unified set of measures nor a standard framework to assess the accuracy of an attack has so far been proposed. The accuracy measures used in existing attack methods are usually ad-hoc. In the following we define a set of measures based on disclosure risks and the evaluations of existing attacks on PPRL to quantify the success of a privacy attack on PPRL [41, 216]. The results of any privacy attack on PPRL are based on two primary outputs.

- **Attribute Reidentification**: This is related to attribute disclosure risk [9, 58] and focuses on reidentifying attribute values in an encoded sensitive database. In attribute reidentification, the adversary is able to reidentify some of the actual QID values encoded in $\mathbf{D}^e$. However, attribute reidentification does not necessarily imply that the adversary is capable of reidentifying the actual real-world entity or entities represented by these records [9]. For instance, if the adversary was able to reidentify 100 records in $\mathbf{D}^s$ with the *FirstName* 'John' and the *LastName* 'Smith', then the adversary cannot distinguish with high confidence which record belongs to which actual 'John Smith' in the real-world. However, it is also worth noting that despite the uncertainty of reidentifying unique entities, attribute reidentification in specific databases (such as databases of cancer patients or criminals) can still lead to breach of privacy.
- **Identity Reidentification**: This is related to identity disclosure risk [9, 58] and focuses on reidentifying actual real-world entities using the attribute reidentification results of a privacy attack. If an adversary is able to reidentify individual entities (people) in an encoded database using an attack then their privacy is compromised. It is important to note that in some situations even with low attribute reidentification accuracy there can be high identity reidentification risk for some entities due to the uniqueness of the combination of their QID values [9]. For instance, if an attack was only successful in reidentifying the *LastName* value of a certain record correctly, and if there is only one real-world entity that has this *LastName* value, then the identity of that person is compromised [189].

Despite the fact that certain QID values, such as year of birth, city, gender, and zipcode, can be more common among a large number of people, the combinations of those QID values can make records unique. For instance, consider a scenario where a database has 100 records with the *YearOfBirth* value '1990' and 500 records with the *City* value 'Queanbeyan', but when combined, there is only one record with *YearOfBirth* '1990' and *City* 'Queanbeyan'. Then the identity of that particular person is disclosed. Sweeney [189] was able to uniquely identify over 85% of the population in the United States using only the combination of values in the QIDs zipcode, gender, and date of birth. Therefore, it is important to understand the disclosure risks of identities associated with attacks on PPRL.

The construction principles of all existing attacks on PPRL are focused on identifying QID values encoded in the encoded database $\mathbf{D}^e$ for individual records. However, most attacks do not focus on reidentifying real-world entities based on the identified QID values. Considering both the aspects, in the following, we define six measures to assess the attribute and identity reidentification accuracies of an attack. Some of these measures are used in the evaluations of existing attacks on PPRL [42, 43, 216], but only in an ad-hoc manner.

1. **Correct 1-to-1 matches**: The number of correct assignments (reidentifications) between encoded and plaintext values/records where one encoded value/record is assigned to only one plaintext value/record.

2. **Correct 1-to-*many* matches**: The number of correct assignments between encoded and plaintext values/records where one encoded value/record is assigned to several plaintext values/records and the correct plaintext value/record is among those assignments.

3. **Correct *many*-to-1 matches**: The number of correct assignments between encoded and plaintext values/records where multiple encoded values/records are assigned to one plaintext value/record, where that plaintext value/record is the correct one for at least one of the encoded values/records. Note that *many*-to-1 matches, where multiple encoded values/records are assigned to one plaintext value, can also be considered as multiple 1-to-1 matches where each encoded value/record is assigned to the given plaintext value/record, because the amount of information the adversary is able gain in both of these situations is the same.

4. **Correct *many*-to-*many* matches**: The number of correct assignments between encoded and plaintext values/records where multiple encoded values/records are assigned to multiple plaintext values/records, and at least one correct plaintext and encoded value/record pair is among those assignments.

5. **Wrong matches**: The number of the wrong reidentifications in the total number of reidentifications.

Table 3.2: Different output settings of 1-to-1, 1-to-*many*, *many*-to-1, and *many*-to-*many* assignments based on the column-wise and row-wise overlaps between the encoded and plaintext databases.

		1-1	1-*many*	*many*-1	*many*-*many*
		One encoded record/value is assigned to one plaintext record/value	**One encoded record/value is assigned to multiple plaintext records/values**	**Multiple encoded records/values are assigned to one plaintext record/value**	**Multiple encoded records/values are assigned to multiple plaintext records/values**
Identity reident.	$\mathcal{E}^p \equiv \mathcal{E}^s$, $A^p \equiv A^s$ (same entities, same attributes)	Scenario 1	Scenario 2	Scenario 3	Scenario 4
	$\mathcal{E}^p \equiv \mathcal{E}^s$, $A^p \cap A^s \neq \emptyset$ (same entities, overlapping attributes)				
	$\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$, $A^p \equiv A^s$ (overlapping entities, same attributes)	Scenario 5	Scenario 6	Scenario 7	Scenario 8
	$\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$, $A^p \cap A^s \neq \emptyset$ (overlapping entities, overlapping attributes)				
Attribute reident.	$\mathbf{Q}^s \equiv \mathbf{Q}^p$ (same values)	Scenario 9	Scenario 10	Scenario 11	Scenario 12
	$\mathbf{Q}^s \cap \mathbf{Q}^p \neq \emptyset$ (overlapping values)	Scenario 13	Scenario 14	Scenario 15	Scenario 16

6. **None-matches**: The number of encoded values/records that had no reidentifications, i.e. the attack was not able to assign any of the plaintext values/records to these encoded values/records.

As we discussed in Section 3.5.3, there can be different column-wise and row-wise overlaps between the databases $\mathbf{D}^p$ and $\mathbf{D}^s$. Depending on these overlaps, the 1-to-1 (1 link), 1-to-*many* (m links), *many*-to-1 (m links), and *many*-to-*many* ($m \cdot m$ links) assignments an attack obtain can mean different output scenarios. In Table 3.2, we show these various scenarios and below we discuss those scenarios in detail with respect to both attribute and identity reidentifications.

- **1-1** | $\mathcal{E}^p \equiv \mathcal{E}^s$, $A^p \equiv A^s$ (Scenario 1): Two possible outcomes: (1) the link is correct where both the encoded and the plaintext records refer to the same real-world entity, and (2) the link is wrong where the record pair refers to two different real-world entities, but both these entities occur in both $\mathbf{D}^p$ and $\mathbf{D}^s$.
- **1-*many*** | $\mathcal{E}^p \equiv \mathcal{E}^s$, $A^p \equiv A^s$ (Scenario 2): Two possible outcomes (from m links): (1) only one link is correct where one of the plaintext records refer to the same real-world entity as the encoded record, and (2) all the links are wrong where none of the plaintext records refer to the same real-world entity as the encoded record. All the entities occur in both $\mathbf{D}^p$ and $\mathbf{D}^s$.
- ***many*-1** | $\mathcal{E}^p \equiv \mathcal{E}^s$, $A^p \equiv A^s$ (Scenario 3): Two possible outcomes (from m links): (1) only one link is correct where one of the encoded records refer to the

same real-world entity as the plaintext record, and (2) all the links are wrong where none of the encoded records refer to the same real-world entity as the plaintext record.

- ***many-many*** | $\mathcal{E}^p \equiv \mathcal{E}^s$, $A^p \equiv A^s$ (Scenario 4): Three possible outcomes (from $m \cdot m$ potential links): (1) a group of encoded records (entities) are exactly the same as the group of plaintext records (entities) so every encoded entity in the encoded records has its corresponding plaintext entity in the plaintext records, (2) only some of the encoded records (entities) are correctly assigned to a group of plaintext records (entities), (3) all the assignments are wrong where there is no overlap between the encoded entities and the plaintext entities (none of the encoded/plaintext record pairs refer to the same real-world entity).

- **1-1** | $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$, $A^p \equiv A^s$ (Scenario 5): Two possible outcomes: (1) the link is correct where both the encoded and the plaintext records refer to the same real-world entity which are common to both the databases, and (2) the link is wrong where the record pair refers to two different real-world entities, which can either occur in both $\mathbf{D}^p$ and $\mathbf{D}^s$ or one entity can occur in $\mathbf{D}^p$ only while the other in $\mathbf{D}^s$ only (and these two entities can have the same values in the encoded attributes).

- **1-*many*** | $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$, $A^p \equiv A^s$ (Scenario 6): Two possible outcomes (from m links): (1) only one link is correct where one of the plaintext records refer to the same real-world entity as the encoded record and this entity is common to both the databases whereas other plaintext entities may or may not occur in the encoded database, and (2) all the links are wrong where none of the plaintext records refer to the same real-world entity as the encoded record. In this scenario both the encoded entity and the plaintext entities may or may not occur in $\mathbf{D}^p$ and $\mathbf{D}^s$, respectively.

- ***many*-1** | $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$, $A^p \equiv A^s$ (Scenario 7): Two possible outcomes (from m links): (1) only one link is correct where one of the encoded records refer to the same real-world entity as the plaintext record and this entity is common to the both databases whereas other encoded entities may or may not occur in the plaintext database, and (2) all the links are wrong where none of the encoded records refer to the same real-world entity as the plaintext record. In this scenario both the plaintext entity and the encoded entities may or may not occur in $\mathbf{D}^p$ and $\mathbf{D}^s$, respectively.

- ***many-many*** | $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$, $A^p \equiv A^s$ (Scenario 8): Three possible outcomes (from $m \cdot m$ potential links): (1) the group of encoded records (entities) are exactly the same as the group of plaintext records (entities) so every encoded entity in the encoded records has its corresponding plaintext entity in the plaintext records, (2) only some of the encoded entities are correctly assigned to a group of plaintext entities, (3) all the assignments are wrong where there is

no overlap between the encoded entities and plaintext entities (none of the encoded/plaintext record pairs refer to the same real-world entity). If a link is correct then the corresponding entity occurs in both $\mathbf{D}^p$ and $\mathbf{D}^s$, while if a link is wrong those entities may or may not occur in both the databases.

- **1-1** | $\mathbf{Q}^s \equiv \mathbf{Q}^p$ (Scenario 9): Two possible outcomes: (1) the link is correct where the value pair must represent the same QID value, and (2) the link is wrong where the value pair represents two different QID values. In both scenarios values occur in both $\mathbf{D}^p$ and $\mathbf{D}^s$.

- **1-*many*** | $\mathbf{Q}^s \equiv \mathbf{Q}^p$ (Scenario 10): Two possible outcomes (from m links): (1) only one link is correct where one of the value pairs refer to the same QID value, and (2) all the links are wrong where none of the value pairs refer to the same QID value.

- ***many*-1** | $\mathbf{Q}^s \equiv \mathbf{Q}^p$ (Scenario 11): Two possible outcomes (from m links): (1) only one link is correct where one of the value pairs refer to the QID same value, and (2) all the links are wrong where none of the value pairs refer to the QID same value.

- ***many-many*** | $\mathbf{Q}^s \equiv \mathbf{Q}^p$ (Scenario 12): Three possible outcomes (from $m \cdot m$ potential links): (1) all of the encoded QID values are correctly assigned to their corresponding plaintext QID values, (2) some of the encoded QID values are correctly assigned to their corresponding plaintext QID values referring to the same real-world entities, (3) all the links are wrong where none of the encoded/plaintext value pairs refer to the same QID value.

- **1-1** | $\mathbf{Q}^s \cap \mathbf{Q}^p \neq \emptyset$ (Scenario 13): Two possible outcomes: (1) the link is correct where the value pair must represent the same QID value and the value occur in both the databases, and (2) the link is wrong where the value pair represents two different QID values and the these values may or may not occur in both the databases.

- **1-*many*** | $\mathbf{Q}^s \cap \mathbf{Q}^p \neq \emptyset$ (Scenario 14): Two possible outcomes (from m links): (1) only one link is correct where one of the value pairs refer to the same QID value, and this QID value is common to both the databases, and other plaintext QID values may or may not occur in the encoded database, and (2) all the links are wrong where none of the value pairs refer to the same QID value, and furthermore, encoded and plaintext QID values may or may not occur in $\mathbf{D}^p$ and $\mathbf{D}^s$, respectively.

- ***many*-1** | $\mathbf{Q}^s \cap \mathbf{Q}^p \neq \emptyset$ (Scenario 15): Two possible outcomes (from m links): (1) only one link is correct where one of the value pairs refer to the same QID value, and this QID value is common to both the databases, and other encoded QID values may or may not occur in the plaintext database, and (2) all the links are wrong where none of the value pairs refer to the same QID value, and

furthermore, encoded and plaintext QID values may or may not occur in $\mathbf{D}^p$ and $\mathbf{D}^s$, respectively.

- ***many-many*** | $\mathbf{Q}^s \cap \mathbf{Q}^p \neq \emptyset$ (Scenario 16): Three possible outcomes (from $m \cdot m$ potential links): (1) all of the encoded QID values are correctly assigned to their corresponding plaintext QID values, (2) some of the encoded QID values are correctly assigned to their corresponding plaintext QID values referring to the same real-world entities, (3) all the links are wrong where none of the encoded/plaintext value pairs refer to the same QID value. If a link is correct then the corresponding QID value occurs in both $\mathbf{D}^p$ and $\mathbf{D}^s$ and if a link is wrong those QID values may or may not occur in both the databases.

Since identity reidentification is directly related to the uniqueness of a record, this uniqueness can also be used to evaluate the accuracy of identity reidentification in an attack. For a given encoded record $r \in \mathbf{D}^e$, assume that $\mathbf{c}$ is the set of attribute values reidentified correctly for that record. The probability of suspicion disclosure risk measure proposed by Vatsalan et al. [200] (as we discussed in Section 2.6.3) uses the uniqueness of attribute values in a plaintext database to assess the risk of reidentification. This measure can be adapted to calculate the probability of suspicion (PS), for an encoded record based on the set of reidentified attribute values $\mathbf{c}$ for that record. The normalised PS value for the record r would be:

$$r_{ps} = \frac{1/n_c - 1/n}{1 - 1/n}, \tag{3.2}$$

where n_c is the number of records in the plaintext database which have the exact same set of reidentified attribute values $\mathbf{c}$, and $n = |\mathbf{D}^p|$ is the total number of records in the plaintext database $\mathbf{D}^p$. Using the normalised PS values for all records in $\mathbf{D}^e$ that each has at least one reidentified plaintext value assigned to it, four aggregated accuracy measures can be defined [200] (see Section 2.6.3 for equations).

1. Maximum risk: Maximum PS value of any records in $\mathbf{D}^e$.
2. Mean risk: Average PS value of all the records in $\mathbf{D}^e$.
3. Median risk: Median PS value of all the records in $\mathbf{D}^e$.
4. Marketer risk: The ratio of the number of records in $\mathbf{D}^e$ with a PS value of 1.

3.7.2 Scalability of an Attack

With increasing amount of data being collected every day, it is important to assess the scalability of an attack in order to analyse how practical the attack can be when conducted on large encoded databases. The practical scalability of an attack can be evaluated based on its runtime and memory consumption. Certain existing attacks on PPRL, as we discuss in Chapter 4, that are based on matrix optimisation or that consists of manual processes will likely not be scalable to very large databases

because of their high memory requirement and long runtimes. However, some attacks that are based on frequency alignments have shown to be scalable to encoded databases with millions of records, as we illustrate in Table 3.3. Apart from the practical analysis of runtime, one can also theoretically analyse the computational complexity of an attack, as we discussed in Section 3.6.4, to determine the worst-case runtime of an attack algorithm.

3.7.3 Implementation of an Attack

Different programming languages have been used to implement existing attacks on PPRL. For instance, as we show in Table 3.3, most existing attacks on PPRL are implemented using Python. It is important to know whether an attack has been implemented and evaluated empirically to understand its real-world applicability. Some attacks proposed in the literature provide necessary source-code to facilitate replication of their experiments [42, 43], while others do not.

It is also important to understand whether an attack is fully automated or it includes manual processes which require human involvement. If an attack consists of manual processes it will likely reduce its practicality in certain real-world situations, especially when applied on large encoded databases. For example, in the attack proposed by Niedermeyer et al. [147], several steps are conducted manually. All the other attacks on PPRL we discuss in Chapter 4 are fully automated attacks.

3.7.4 Databases Used for Evaluation

An attack on PPRL can be evaluated using real or synthetic databases. In order to conduct a critical evaluation of an attack, ideally multiple databases with various real-world data characteristics, such as different QID combinations, records with missing values, and attributes with errors, are to be used. However, because it is difficult to always obtain real databases with personally identifiable information [40], synthetic databases with similar characteristics can also be used in the evaluation process. A comprehensive evaluation of an attack using a set of databases will provide a better understanding of how well it will perform in different real-world situations. This will allow DOs to take necessary precautions within a PPRL protocol to prevent possible reidentification of values from such attacks.

We now analyse and categorise existing attacks on PPRL using our proposed taxonomy in Table 3.3. In this table, we use the following abbreviations: SE - Social engineering, HM - Human mistakes, RP - Use of recommended parameters; MSK - Masking, BF - Bloom filters, TMH - Tabulation min-hashing, 2SH - Two-step hashing, MMK - Multiple match-keys, PTB - Perturbation, HMAC - Keyed-hash message authentication code, ST - Similarity tables; NCVR - North Carolina voter registration database, GN - German names, GNA - German names and addresses, ALN - Australian last names, MVR - Michigan voter registration database, UKCD - Rawtenstall (England) census database, TITANIC - Titanic passenger database, and EURO-CS - European census database (synthetic).

Table 3.3: A characterisation of existing privacy attacks on different encoding techniques using the twelve dimensions illustrated in Figure 3.2. The abbreviations used are described on page 61. For complexity analysis we assumed $n = |\mathbf{D}^e| = |\mathbf{D}^p|$, $\mathbf{q}$ = the set of all q-grams from the QID values in $\mathbf{D}^p$, and $m = |A^p| = |A^s|$. Note that last three attack methods correspond to the ones we propose in this thesis from Chapters 6 to 8.

Attack method	Adversarial aspects				Technical aspects				Practical aspects			
	Adversary type	**Non-tech. issues**	**Encoding param.**	**Data assumptions**	**Attack type**	**Attack scope**	**Exploited vulnerability**	**Complexity**	**Success** (# reident.)	**Scalability** (runtime)	**Implement.** (language)	**Eval.** (data)
Kuzu et al. (2011)	IA, OA	SE/HM	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \supset \mathcal{E}^s$	Cryptanalysis	BF	Frequency, Length, Co-occurrence	$O(2^n)^*$	400 out of 3,500 values	$>$ 20 min	Java	NCVR
Niedermeyer et al. (2014)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \supset \mathcal{E}^s$	Cryptanalysis	BF	Frequency	$O(n)^*$	934 out of 10,000 values	multiple days	R	GN
Kroll and Steinmetzer (2015)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \supset \mathcal{E}^s$	Cryptanalysis	BF	Frequency, Co-occurrence	$O(n + \lvert\mathbf{q}\rvert!)^*$	44,000 out of 100,000 records	$>$ 402 min	C++/ Python	GNA
Christen et al. (2017)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$	Cryptanalysis	BF	Frequency	$O(n \cdot log\ n)$	6 to 10 out of top 100 values	$<$ 2 sec	Python	NCVR UKCD
Mitchell et al. (2017)	IA	SE/HM/RP	$\mathbf{p}' \equiv \mathbf{p}$, s	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \equiv \mathcal{E}^s$	Dictionary	BF	Co-occurrence	$O(n)$	364,450 out of 474,319 values	395 min	C++	NCVR
Culnane et al. (2017a)	IA, OA	HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \subset \mathcal{E}^s$	Similarity	HMAC and ST	Similarity neighbourhood	$O(n^2)^*$	93% out of 390,523 values	$<$ 1 day	Java	ALN
Christen et al. (2018a)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$	Cryptanalysis	BF	Frequency, Length	$O(n^2)$	41 to 27,665 out of 222,251 values	$<$ 22 min	Python	NCVR MVR, UKCD
Christen et al. (2018b)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$	Cryptanalysis	BF	Frequency, Co-occurrence	$O(n^2)$	10 to 5,000 out of 222,251 records	$<$ 150 min	Python	NCVR
Vidanage et al. (2019) (Chapter 7)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$	Cryptanalysis	BF	Frequency, Co-occurrence	$O(n^2)^*$	163 to 36,796 out of 222,251 records	$>$ 1 day	Python	NCVR
Vidanage et al. (2020a) (Chapter 6)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \supset \mathbf{E}$ $\mathcal{E}^p \cap \mathcal{E}^s \neq \emptyset$	Frequency	MMK	Frequency	$O(2^m \cdot n)^*$	14,415 out of 8,114,702 records	$<$ 210 min	Python	NCVR MVR
Vidanage et al. (2020b) (Chapter 8)	IA, OA	SE/HM/RP	$\mathbf{p}' \subset \mathbf{p}$	$\mathbf{Q}^p \equiv \mathbf{E}$ $\mathcal{E}^p \equiv \mathcal{E}^s$	Similarity	BF, TMH, 2SH	Similarity neighbourhood	$O(n^2)^*$	55.1% out of 25,343 records	$>$ 1 day	Python	NCVR, TITANIC, EURO-CS

$*$ estimated from the technical descriptions of the attacks.

3.8 Summary

In this chapter we have presented a novel taxonomy of attacks on privacy-preserving record linkage (PPRL). We have identified twelve dimensions that can be used to characterise privacy attacks on PPRL, and have discussed each of these dimensions comprehensively focusing on important aspects such as adversary types, assumed knowledge of the adversary, different attack types, the vulnerabilities exploited by attacks, and measuring the success of attacks on PPRL.

This taxonomy will be beneficial for the owners of sensitive database to evaluate and compare different attacks on PPRL in order to identify the strengths and limitations of these attacks, and whether they are successful on their sensitive databases. To the best of our knowledge, no such taxonomy currently exists in the literature. However, we acknowledge that, given most existing attacks on PPRL can only be applied on certain PPRL techniques, it would be difficult to conduct an experimental evaluation of all attacks with regard to their applicability on different PPRL encoding and encryption techniques.

Chapter 4

Existing Attacks on Privacy-Preserving Record Linkage

In this chapter we provide a comprehensive analysis of existing attack methods on privacy-preserving record linkage (PPRL) techniques. In Section 4.1 we give a brief introduction to privacy attacks on PPRL highlighting the importance of analysing such attack methods. In Section 4.2 we then outline seven existing attack methods proposed on PPRL by describing the employed techniques and assumptions made, their reidentification accuracy, and the limitations of these attacks. In Section 4.3 we then discuss three different attack methods that are not directly related to PPRL but are conducted on real-world databases to reidentify anonymised sensitive data. Finally, in Section 4.4 we provide a summary of our analysis.

4.1 Introduction

Out of the two main categories that privacy-preserving record linkage (PPRL) techniques can be divided into (as we discussed in Chapter 2), perturbation based techniques have shown to be susceptible to certain attacks. This is due to the different vulnerabilities in perturbation based techniques that can be exploited by attacks. However, all except one of these existing attack methods have been proposed for Bloom filter (BF) encoding for PPRL [177]. While BF encoding can successfully be attacked when used with specific combinations of encoding parameter settings (as we describe next), these attack methods also have limitations, such as excessive time and memory consumption, and the requirement of specific knowledge of the encoding parameters. Such shortcomings make some of these attack methods impractical to be applied on real-world encoded sensitive databases.

In Chapter 3 we proposed a taxonomy of attacks on PPRL where we categorised existing attacks using twelve dimensions, and in Table 3.3 on page 62 we summarised the characteristics of each attack method using those dimensions. The aim of this chapter is to further analyse each of the existing attack method on PPRL based on our proposed taxonomy in order to better understand the different methodologies employed by these attacks and how they are evaluated, and identify the limitations.

Table 4.1: Specific notation used throughout this chapter. For common notation used throughout the thesis, refer to the list of notation at the beginning of the thesis.

$\mathbf{A}$	Atom correlation matrix
$\mathbf{P}$	Q-gram correlation matrix
$\mathbf{C}^P$	List of potential q-gram sets for all bit positions
$\mathbf{C}^N$	List of not possible q-gram sets for all bit positions
$\mathbf{C}^A$	List of assigned q-gram sets for all bit positions
$\mathbf{c}_p^+$	Set of possible q-grams for the position p
$\mathbf{c}_p^-$	Set of not possible q-grams for the position p
$\mathbf{c}_p$	Set of potential q-grams for the position p where $\mathbf{c}_p = \mathbf{c}_p^+ \setminus \mathbf{c}_p^-$
$\mathbf{v} = \{v_1, v_2, ...v_i\}$	Set of variables (quasi-identifying (QID) attribute values)
$\mathbf{c} = \{c_1, c_2, ...c_j\}$	Set of constraints
$\mathbf{a}$	An atom
$\mathbf{q}$	Set of q-grams derived from a QID value s
q	A q-gram where $q \in \mathbf{q}$
k	Number of hash functions used for the BF encoding
s	A QID value
f^l, f^h	Lower and upper bounds of a frequency interval

Understanding the characteristics and drawbacks of existing attack methods will help us to develop novel and more sophisticated attack methods on different PPRL techniques as we propose in Chapters 6 to 8.

4.2 Existing Privacy Attacks on PPRL

Over the last decade eight privacy attacks have been proposed for PPRL. In the following we describe and analyse seven of those eight attacks. To avoid repetition, we exclude the eighth attack initially proposed by Christen et al. [43] from this discussion as we propose an extension for this attack in Chapter 7 and the methodology of the attack will be discussed in that chapter. For each of the seven attacks, in the following we describe the assumptions made, the employed methodology, the evaluation criteria, and the limitations of the attack. Specific notation used throughout this chapter are listed in Table 4.1.

4.2.1 Constraint Satisfaction based Cryptanalysis for Bloom Filters

The first attack on a PPRL technique was proposed by Kuzu et al. [121] in 2011 for BF encoding. The attack was based on a frequency aware constraint satisfaction problem (CSP) solver.

Preliminaries and Assumptions: The attack assumes that the record linkage (RL) process is conducted using a three party protocol where a linkage unit (LU), as we discussed in Section 2.4, is employed to perform the linkage of encoded values. The

LU is considered as an adversary who does not collude with any of the database owners (DOs), but attempts to break the encoding and reidentify encoded quasi-identifying (QID) attribute values using the information available to her. The proposed attack is based on two main assumptions: (1) the sensitive database, $\mathbf{D}^s$, is a random sample derived from a global population, $\mathbf{G}$[1], such that $\mathbf{D}^s \subseteq \mathbf{G}$, and the adversary has access to $\mathbf{G}$, and (2) the adversary knows the number of hash functions, k, that was used for the BF encoding. The authors [121] argued that the adversary is able to estimate k by analysing the distributions of the numbers of 1-bits in each BF in an encoded database $\mathbf{D}^e$.

Methodology: The proposed attack can be divided into three parts: a frequency analysis, a BF encoding analysis, and the CSP solver.

1. **Frequency Analysis**: Initially a frequency alignment is used to map frequent plaintext QID values in $\mathbf{G}$ with frequent BFs in the encoded database $\mathbf{D}^e$. When modelling the CSP, candidate QID values are considered as *variables* and candidate BFs are considered as *domains*. Using a suitable sampling technique [143], multiple random samples of size $|\mathbf{D}^e|$ are derived from $\mathbf{G}$ and the frequencies of each unique variable (a QID value) in those samples are calculated. These frequencies are then used to calculate frequency intervals (lower and upper bounds), f^l and f^h, for each unique variable. The CSP is designed to contain only the variables with lower bound $f^l > 0$. Then each domain (BF) in $|\mathbf{D}^e|$ is assigned to a variable (QID value) by mapping domain frequencies with variable frequency intervals. For instance, a BF $\mathbf{b}_i$ is assigned to variable v_i if and only if $f^l_{v_i} \leq f_{\mathbf{b}_i} \leq f^h_{v_i}$. Due to noise in $\mathbf{G}$ (typographical errors and missing values), these frequency intervals might contain errors. To reduce that, the authors have proposed to calculate possible error rates, err_l and err_h, for each frequency interval using domain expert or information from historical data.

2. **BF Encoding Analysis**: A set of constraints are decided and assigned to the CSP solver where a CSP is defined by a set of variables $\mathbf{v} = \{v_1, v_2, ...v_i\}$ and a set of constraints $\mathbf{c} = \{c_1, c_2, ...c_j\}$. Each variable v_i has a non-empty set of domains (BFs) assigned to it. Each constraint c_j is mapped to one or more variables in $v_i \in \mathbf{v}$ allowing only certain assignments of domains to those variables possible. Constraints are defined by analysing the behaviour of the encoding function, $encode()$, and the number of hash functions, k. For instance if a variable v_i has domains with a common bit position set to 0, that means none of the q-grams $q \in \mathbf{q}_{v_i}$ in that variable can set that position to 1 where $\mathbf{q}_{v_i}$ is the set of q-grams derived from the variable v_i.

   A BF of length l is generated for each possible q-gram in the considered alphabet. Initially, the BF $\mathbf{b}_{q_i}$ assigned to a q-gram q_i has all its bit positions set to 1. With assignment of domains to variables in the CSP, these BFs of individual q-grams are updated. With an assignment of a domain $\mathbf{b}_j$ to the variable v_j, a bit-wise AND ($\wedge$) operation is applied to BFs $\mathbf{b}_{q_i}$ of all q-grams $q_i \in v_j$ such

[1]This is defined as a global dataset in the original work [121]

that $\mathbf{b}_{q_i} = \mathbf{b}_{q_i} \wedge \mathbf{b}_j$. For instance, if the domain [0110101011] is assigned to the variable 'smith', the initial BF $\mathbf{b}_{sm} = [1111111111]$ of the q-gram 'sm' is updated as $\mathbf{b}_{sm} = [0110101011] \wedge [1111111111] = [0110101011]$. Here it is important that the ground truth is not altered when bit-wise AND is applied for each $\mathbf{b}_{q_i}$ such that for all $q_i \in v_j$, the following must hold every time:

$$\bigvee_{i=1}^{n} \mathbf{b}_{q_i} = \mathbf{b}_{v_i}, \tag{4.1}$$

where n is the number of q-grams in v_i. A threshold, w, is defined to set the minimum number of bit locations set by each q-gram q_i. If the threshold, w, is set to a high value, that could reduce the search space of the CSP significantly. However, this may lead to false positives depending upon the collision probability of collisions in bit positions.

3. **CSP Solver**: Once the problem is modelled by assigning candidate domains to variables, and setting up relevant constraints, the CSPs are solved by applying a standard backtracking search algorithm [170]. In order to improve the efficiency of the search the authors also proposed to use additional heuristics [121]. One such heuristic is the *dom/deg* ratio, which selects the variable with the smallest *domain* involved in the greatest number of constraints (*degree*). If a variable contains frequently used q-grams then most of the constraints contain that variable. If the domain of that variable is small then it can be solved through the CSP solver with high precision. For a given variable v_i, its *dom/deg* can be defined as:

$$dom/deg_{v_i} = \frac{|domain_{v_i}|}{\sum_{j=1}^{m} f_{q_j}}, \tag{4.2}$$

where $|domain_{v_i}|$ is the size of the domain assigned to v_i, $m = |\mathbf{q}_{v_i}|$ is the number of q-grams in v_i, and f_{q_j} is the frequency of the q-gram q_j in all variables of the CSP. For backtracking search the variable with the smallest *dom/deg* is initially selected, then the second smallest is selected next, and so on [121].

Experimental Evaluation: The proposed attack was evaluated using a North Carolina Voter Registration (NCVR) database and the attack was able to reidentify 400 frequent encoded first names from 3,500 unique BFs accurately. The authors [121] have observed that the recall and precision of the results increases with the number of variables assigned to each constraint c_i. When the threshold w increased the complexity of the attack is decreased due to a reduction of the search space of the CSP. In a followup work, Kuzu et al. [122] carried out another set of experiments to evaluate the proposed attack using a real patient database, where the attack was successful in reidentifying 4 out of 20 frequent encoded first names correctly.

Limitations and Discussion: The attack assumes that the sensitive database, $\mathbf{D}^s$ (encoded as $\mathbf{D}^e$), is a complete random sample from the global population, $\mathbf{G}$, which

is also publicly accessible. In real-world scenarios, however, an adversary may not have access to such a global population and $\mathbf{D}^e$ may not be a complete random sample of $\mathbf{G}$ as we discussed in Chapter 3. The experimental results showed that with the increasing number of variables assigned to each constraint the runtime of the attack increases and when the length of q-grams increases the complexity of the attack grows. Both these observations illustrate the non-scalability of the attack. In the experimental evaluation conducted on a patient database [122], it was shown that the proposed attack is not successful when multiple QID values are encoded into a single BF using encoding techniques such as record-level BF (RBF) [61] or cryptographic long-term key (CLK) [178], two encoding techniques we describe in detail in Appendix A.3.

4.2.2 Frequency based Manual Cryptanalysis for Bloom Filters

The second attack method on PPRL (also for BF encoding) was proposed by Niedermeyer et al. [147]. This attack was based on a database of frequent German last names. This is a manual attack where each step of the attack should be carried out by an adversary based on the results of the previous steps.

Preliminaries and Assumptions: As with the previous attack on BF encoding [121], this attack considers a three party protocol where a LU, a semi-trusted third party, acts as the adversary and attempts to reidentify encoded QID values from the database $\mathbf{D}^e$. It is assumed that the adversary does not collude with any of the DOs and that it has access to the frequency information of QID values that are being encoded such as a dataset of German last names. Furthermore, it is also assumed that the adversary knows the number of hash functions, k, that was used for BF encoding, and that the hashing of q-grams was done using the double hashing (DH) [177] technique we describe in Appendix A.

Methodology: The attack consists of five main steps. In the first step BFs in the encoded database, $\mathbf{D}^e$, are deduplicated and sorted according to their frequencies. From 10,000 BFs in the testing dataset, 7,580 unique BFs were extracted. From those 7,580 unique BFs 934 BFs have been filtered as occurring at least twice in the database, $\mathbf{D}^e$. The attack was only applied on these 934 BFs in order to prevent any error which might occur due to rare last name values or q-grams [177]. Since it is assumed that the adversary knows about the DH technique and the used number of hash functions, k, in the second step all possible BFs which can be generated by encoding a single q-gram are constructed. For a given q-gram q, the encoding function $encode()$ can output any bit pattern, $encode(q) \longrightarrow \{0,1\}^l$, of length l. But with the DH technique there can only be a limited number of 1-bit patterns when hashing a single q-gram. This is because the equation used for DH, $[h_1(q) + i \cdot h_2(q)] \ mod \ l$, can only output a certain set of bit patterns, where h_1 and h_2 are the two hash functions used, $1 \leq h_1(q_i),\ h_2(q_i) \leq l$, and $1 \leq i \leq k$. The authors referred to these patterns as atoms [147], and for BFs of length $l = 1,000$, there can be 1,000,000 possible candidate atoms. From all possible candidate atoms, 3,952 possible atoms were selected

by comparing them with BFs from the previous step. In the third step, the attack filters out atoms which do not have a Hamming weight (the number of 1-bits) of k. This is valid because it is assumed that the adversary's knows the used number of hash functions, k. However, with this filtering process there is a possibility of removing atoms that are encoded in BFs, but have a Hamming weight less than k due to collisions. But the authors have shown that the probability of having such atoms are fairly low (with $k = 15$ the probability is 0.016) [147]. Using this filtering process 805 candidate atoms were selected out of 3,952 atoms.

In the fourth step, the attack determines the frequency of each atom occurring in the BFs from the first step. The atoms are then sorted in descending order according to their frequencies to reduce the influence of wrongly assigned atoms in the fifth step. In the final step, using different lists of QID values (in this attack a list of German last names) the most frequent plaintext q-grams are identified. The authors have used Wikipedia as one of their source for lists of the most frequent German last names[2] in order to find frequent q-grams. These q-grams are then assigned to the candidate atoms based on their frequencies. Using the q-gram assignments and the knowledge of German last names and their frequencies the adversary can then guess the possible QID values encoded into each BF manually. The last step of the attack highly depends on the adversary's knowledge of the domain of German last names. For instance the string 'sch' is a common German prefix and therefore the q-gram 'ch' is one of the most frequent q-grams in German names. Hence, the corresponding atom of the q-gram 'sc' can be determined by analysing which atom generally occurs together with the atom assigned to q-gram 'ch'. A detailed description of this manual process can be found in [147].

Experimental Evaluation: From 7,580 unique German last names encoded into 10,000 BFs, the attack was able to reidentified the 934 most frequent last names in the experimental evaluation. A statistical analysis was conducted in order to verify the accuracy of the assignment of last name values to BFs. In this analysis 700,000 German last names were encoded using two arbitrary different hash functions and the Hamming weight distribution was analysed against the Hamming weight distribution of the Bloom filters used in the attack. The results of this analysis showed that the two distributions are approximately similar (limited details about the frequency analysis were given in the paper).

Limitations and Discussion: The attack assumes that the adversary has access to the frequency information of German last names and has knowledge about the DH technique [177] as well as the number of hash functions used in the BF encoding process. In a real-world scenario this is a highly unlikely situation since the encoding parameters are not usually revealed to the LU. The final step of the attack, where each last name was assigned to a BF, was a manual process which needs be carried out by the adversary. The proposed attack also works best without any hardening methods applied and with only one attribute encoded into BFs.

[2] https://en.wikipedia.org/wiki/List_of_the_most_common_surnames_in_Germany

4.2.3 Optimisation based Cryptanalysis for Bloom Filters

The frequency based manual cryptanalysis proposed by Niedermeyer et al. [147], described in the previous section, was extended by Kroll and Steinmetzer [119] into a fully automated cryptanalysis of BFs that can have multiple QID values encoded into them. This attack was conducted on German last names, first names, and birth locations. The proposed attack employed an optimisation function in order to build a similarity vector between q-grams and BFs.

Preliminaries and Assumptions: Similar to the previous attacks, this attack considered a three-party protocol where a LU (semi-trusted third party) acts as the adversary to infer as much knowledge as possible from the information available to her. The attack assumes that the adversary has access to frequency information of German first names, last names, and birth locations, i.e. the sensitive database, $\mathbf{D}^s$, is a random sample drawn from a publicly available global population, $\mathbf{G}$. Furthermore it is also assumed the adversary knows the used number of hash functions, k, and that the DH technique has been used in the BF encoding.

Methodology: In the initial step of the attack, all possible bit patterns of BFs of length l which can be created based on DH technique [177] are constructed, as we described in the previous section. Each of these bit patterns refers to the encoding of a single q-gram which is called an atom. For each atom, a set of BFs, for which all the 1-bit positions in the atom are set to 1, is selected from the encoded database $\mathbf{D}^e$. Then for each selected set of BFs the bit-wise AND operation is applied to get a single BF for the corresponding atom. For instance if atom $\mathbf{a}_i$ has the set of BFs, $\{\mathbf{b}_1, \mathbf{b}_2, ... \mathbf{b}_n\}$ assigned to it, a single BF, $\mathbf{b}_i$, is derived using $\mathbf{b}_i = \bigwedge_{j=1}^{n} \mathbf{b}_j$. If $\mathbf{b}_i \equiv \mathbf{a}_i$, then the atom $\mathbf{a}_i$ is considered to be a true atom of a q-gram (an atom that actually encodes a q-gram). The resulting set of atoms are then further reduced by keeping only the atoms which have a Hamming weight of 8, 10, or 20 because $k = 20$ hash functions are used to encode QID values into BFs. All the other atoms are discarded. After the above steps, the authors were able to identify 1,776 true atoms using 100,000 encoded BFs [119] in $\mathbf{D}^e$.

In the next step of the attack, atoms were sorted in descending order according to their frequencies of occurring in BFs, and for each BF a set of atoms is assigned based on the occurrence of atoms in BFs. Then, assuming the adversary has knowledge about the frequency information of German last and first names, and birth locations, the most frequent q-grams are identified and assigned to corresponding atoms. It is also assumed that when encoding multiple QID values into a single BF, q-grams from each attribute are combined with a suffix or a prefix depending on the QID before hashing. For instance a q-gram 'er' from a last name can be suffixed as 'er*l*' while if the q-gram is from a first name it can be suffixed as 'er*f*'. Hence the same q-gram is hashed differently when it occurs in different QIDs.

The authors argued that results gained from analysing the relationships between single q-grams and atoms is not enough for automatic reidentification because, except for a few frequent q-gram and atom combinations, the frequencies of q-gram/atom pairs are too close together to allow such kind of assignments. Therefore, in the

next step, the attack calculates and compares the correlations between q-gram pairs and the correlations between atom pairs.

The attack models the correlation between q-grams and atoms into two matrices, **P** and **A**, respectively. These matrices are two dimensional and hold the ratios that two q-grams or two atoms appear with each other with respect to the total number of records in each database. For instance, if atoms $\mathbf{a}_i$ and $\mathbf{a}_j$ occur together in m records (BFs), their respective cell $\mathbf{A}_{i,j}$ in the matrix **A** will have the value $m/|\mathbf{D}^e|$ where $|\mathbf{D}^e|$ is the total number of BFs in the encoded database. After both the **P** and **A** matrices are formed, an optimisation algorithm [100] is applied to iteratively swap the elements in the matrix **A** in a predefined manner. This swapping is conducted until the calculated difference between the two matrices is minimised and the atoms in matrix **A** have the best possible correlation (mapping) with q-grams in matrix **P**.

In the final step of the attack, q-grams are assigned to atoms based on the optimised matrices from the previous step. Then, for each BF, using the q-grams assigned to it, the attack first separated the q-grams for each QID first and last name, and birth location. This is possible because q-grams are either prefixed or suffixed with a character specific to each QID as we discussed above. Finally, the actual first and last name, and birth location values are reidentified using the assigned q-grams and a reference list of name and birth location values from a plaintext dataset.

Experimental Evaluation: In the evaluation of the attack, the authors [119] were able to reconstruct 59.6% of first names, 73.9% of last names, and 99.7% of birth locations correctly. 44% of values from 100,000 BFs were recovered successfully. To attack these 100,000 BFs, the attack took 402 minutes.

Limitations and Discussion: Similar to the previous attack, this attack works only when the DH technique [177] is used for BF encoding. Furthermore the attack assumes that the adversary knows the used number of hash functions, k, and she has access to the global population, **G**, which has similar frequency information to the encoded database. These assumptions might not be practical in real-world scenarios. Even though the attack is defined as to be able to work when multiple QID values are encoded into a single BF, as mentioned above, q-grams are suffixed based on their QID which make them unique to each QID. However, it is not clear how successful the attack will be when the same q-gram from different QID values is not suffixed or prefixed and therefore considered as the same q-gram.

4.2.4 Frequency based Cryptanalysis for Bloom Filters

Christen et al. [42] proposed a frequency based attack on BF encoding for PPRL which is one of the most practical and efficient attacks developed so far. This attack explores the BF construction principle of hashing sets of elements into bit positions in order to evaluate the possibility of reidentifying the encoded QID values in BFs. Figure 4.1 illustrates the main steps of the attack using an example.

Preliminaries and Assumptions: Unlike the previously proposed attacks, this attack method is independent of the encoding function and its parameters used. The attack

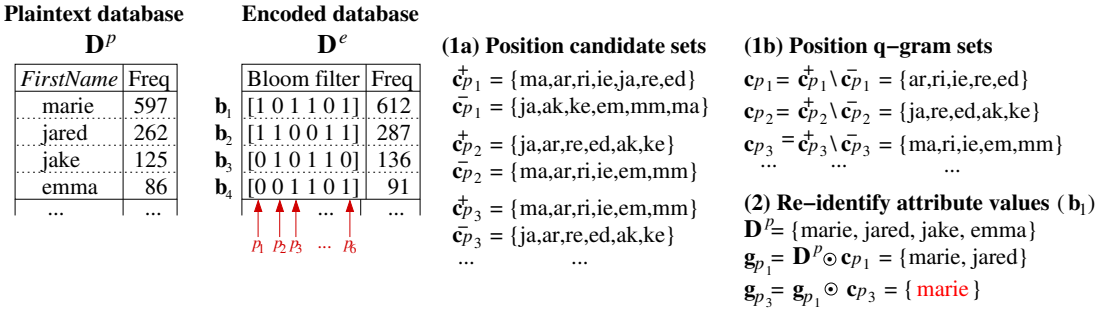


Figure 4.1: Outline of the cryptanalysis attack proposed by Christen et al. [42] (adapted from [42]), which is based on a set of frequent BFs and a set of frequent QID values. In step 1, the attack aligns BFs with plaintext QID values based on their frequencies, to obtain the sets of possible, $\mathbf{c}^+_{p_i}$, and not-possible, $\mathbf{c}^-_{p_i}$, q-grams for each bit position p_i, and then generates sets of potential q-grams, $\mathbf{c}_{p_i}$, for that bit position. In step 2, the attack reidentifies QID values encoded into each BF based on these potential q-gram sets.

assumes that the adversary has access to a database of encoded BFs, $\mathbf{D}^e$, and their frequencies, but does not know anything about the parameters used in the encoding process. It is also assumed that the adversary has access to a publicly available plaintext database, $\mathbf{D}^p$, and the frequencies of the plaintext QID values in this database. Furthermore, the attack assumes that the adversary can guess which QID value(s) have been encoded into BFs by analysing the Hamming weight distribution of BFs in $\mathbf{D}^e$, because for every QID or QID combination a BF database is likely to provides a distinctive distribution of 1-bits [180].

Methodology: The attack consists of two main steps. In the first step, it aligns frequent BFs from $\mathbf{D}^e$ with frequent QID values from $\mathbf{D}^p$ by comparing their frequencies, i.e. the most frequent BF is assigned to the most frequent QID value, the second most frequent BF is assigned to the second most frequent QID value, and so on. At the end of this step the adversary will have a set of BF/QID value pairs ranked by their frequencies. Then each pair of BF/QID values is analysed to find a set of possible q-grams $\mathbf{c}^+_{p_i}$ and a set of not possible q-grams $\mathbf{c}^-_{p_i}$ for each bit position p_i where $1 \leq i \leq l$, and l is the length of a BF. If a certain bit position p_i is set to 1 in a BF, then all the q-grams from the corresponding QID value are added to $\mathbf{c}^+_{p_i}$ while if the bit position is set to 0 then all the q-grams from the QID value are added to $\mathbf{c}^-_{p_i}$. Here the attack uses the BF construction principle which states that if a bit position is set to 1 in a BF at least one of the q-grams from a q-gram set must have been hashed to that position. Similarly, when a bit position is set to 0 then no q-gram from the q-gram set could have been hashed to that BF position.

After obtaining two sets of possible and not-possible q-grams for each BF bit position, the attack generates a set of potential q-grams, $\mathbf{c}_{p_i} = \mathbf{c}^+_{p_i} \setminus \mathbf{c}^-_{p_i}$, that could have been hashed into each bit position p_i. For instance, as illustrated in Figure 4.1, BF bit position p_1 gets the potential q-gram set $\mathbf{c}_{p_1}$ = {ar, ri, ie, re, ed} by applying the set minus operation on the corresponding two sets, $\mathbf{c}^+_{p_1} \setminus \mathbf{c}^-_{p_1}$. At the end of the first

step the adversary obtains a list of such potential q-gram sets, $\mathbf{C} = [\mathbf{c}_{p_1}, \mathbf{c}_{p_2}, ... \mathbf{c}_{p_l}]$ for each bit position in the BFs.

In the second step, the attack attempts to reidentify QID values encoded into each BF using the potential q-grams sets in $\mathbf{C}$ for each bit position. For each BF $\mathbf{b} \in \mathbf{D}^e$ the attack checks its bit positions and based on the value in each bit position (either 0 or 1), the attack removes not possible candidate QID values from a set of frequent QID values from the plaintext database. As shown in Figure 4.1 for $\mathbf{b}_1$, the values 'jake' and 'emma' are not possible because in order to set the bit position p_1 to 1 the value that was encoded into $\mathbf{b}_1$ must contain any one of the q-grams from the set $\mathbf{c}_{p_1}$ = {ar, ri, ie, re, ed}. Similarly, the value 'jared' is also not possible because in order to set the bit position p_3 to 1 the value must contain one of the q-grams from the set $\mathbf{c}_{p_3}$ = {ma, ri, ie, em, mm}. Using the same approach the attack can identify the frequent QID values encoded into each frequent BF in $\mathbf{D}^e$.

Experimental Evaluation: The evaluation of the attack was conducted using two databases, the NCVR [42] database and a census database from the town of Rawtenstall in England (UKCD) [42]. The results showed that the attack was able to reidentify 7 to 10 out of the 10 to 100 most frequent encoded first name values with high precision within 2 seconds (on a server with 64-bit Intel Xeon 2.4 GHz CPUs and with 128 GBytes of memory). The results further illustrated that the proposed attack works even when the hardening techniques, XOR-folding [181] or Balancing [180] as we discuss in Section A.4 in Appendix A have been applied.

Limitations and Discussion: According to a comparison of the proposed attack against the previous attacks presented in the paper [42], the proposed attack is more efficient and more effective having higher accuracy and less computational requirements compared to previous attacks. However, the attack assumes that the frequency information of the encoded database, $\mathbf{D}^e$, is similar to the frequency information of the plaintext database, $\mathbf{D}^p$. Furthermore, the attack only works when the encoded database contains BFs with high and unique frequencies. Such frequency distributions are possible with attribute-level BF encoding (ABF) [177]. With encoding techniques such as CLK [178] or RBF [61], where multiple QID values in a single record are encoded into one BF, the proposed attack will be less successful due to having more unique BFs and therefore less frequency information in $\mathbf{D}^e$.

4.2.5 Extended Frequency based Cryptanalysis for Bloom Filters

An extension of the frequency based cryptanalysis attack by Christen et al. [42] was proposed by the same authors [41] in 2018. Christen et al. [41] have improved the accuracy of their previous attack by analysing the co-occurrences of bit positions and q-grams (using sub- and supersets of frequent QID values and BFs) to expand the sets of q-grams that can be identified for each bit position.

Preliminaries and Assumptions: As with the previous attack [42], this attack method does not require knowledge of the encoding function nor the parameters used for the encoding. The attack assumes that the adversary has access to a BF encoded database,

$\mathbf{D}^e$, and a plaintext database, $\mathbf{D}^p$, where $\mathbf{D}^p$ has a similar frequency distribution of QID values to the frequency distribution of BFs in $\mathbf{D}^e$. The attack also assumes that the adversary can guess what type of QID values have been encoded into BFs by analysing the Hamming weight distribution of BFs in $\mathbf{D}^e$.

Methodology: The proposed attack consists of four main steps. The first step of the attack is similar to the first step of the previous attack [42] where for each bit position p_i in BFs a set of possible, $\mathbf{c}^+_{p_i}$, and a set of not possible, $\mathbf{c}^-_{p_i}$, q-grams are identified by aligning most frequent QID values in $\mathbf{D}^p$ to the most frequent BFs in $\mathbf{D}^e$. Then, unlike the previous attack, two lists of q-gram sets, $\mathbf{C}^P$ and $\mathbf{C}^N$, are derived using the sets of possible and not possible q-grams. The list $\mathbf{C}^P$ contains sets of potential q-grams for each bit position generated as $\mathbf{C}^P[p_i] = \mathbf{c}^+_{p_i} \setminus \mathbf{c}^-_{p_i}$, whereas the list $\mathbf{C}^N$ simply contains the sets of not possible q-grams, $\mathbf{c}^-_{p_i}$, for all bit positions.

In the second step of the attack a list of assigned q-gram sets, $\mathbf{C}^A$, is derived for each bit position using the list $\mathbf{C}^P$ of potential q-grams. For each frequent BF/QID value pair $(\mathbf{b}_i, s_i)$ (obtained based on frequency alignment), the attack first generates the set of q-grams $\mathbf{q}_i$ for the QID value s_i. Next, the attack iterates through every bit position in the BF $\mathbf{b}_i$ and if a bit position p_j is set to 1, then the q-gram set $\mathbf{q}_i$ is intersected with the potential q-gram set, $\mathbf{C}^P[p_j]$, of that position where $1 \leq j \leq l$ and l is the length of a BF. If the resulting q-gram set has only one q-gram, then it is added to the set of assigned q-grams, $\mathbf{C}^A[p_j]$, of that position. This step is used to identify what q-grams has actually been encoded into each bit position out of all potential q-grams.

In the third step, the attack expands the sets of potential, not possible, and assigned q-grams in their corresponding lists by adding additional q-grams from not frequent QID values. First, for each frequent BF/QID value pair, $(\mathbf{b}_i, s_i)$, identified in the first step, the attack extracts not frequent QID values where the q-gram set of a not frequent QID value is either a proper subset or a proper superset of the q-gram set of the frequent QID value s_i. Similarly, not frequent BFs are also extracted where a not frequent BF has either some of the bit positions from $\mathbf{b}_i$ set to 1, or it has all of the bit positions from $\mathbf{b}_i$ along with some additional bit positions set to 1. These shorter (subsets) and longer (supersets) BFs and QID values identified for each frequent pair $(\mathbf{b}_i, s_i)$ are then used to refine existing q-grams in potential, not possible, and assigned q-gram sets, and identify additional q-grams for each bit position using a set of rules. For instance, if a bit position is set to 0 in all not frequent and shorter BFs but it is set to 1 in the corresponding frequent BF $\mathbf{b}_i$, then a q-gram that only occurs in the frequent QID s_i but not in all shorter QID values should be encoded in that position. Similarly, if a bit position in the frequent BF $\mathbf{b}_i$ is set to 0 and in all not frequent and longer BFs that bit position is set to 1, then a common q-gram from longer QID values that does not occur in the frequent QID value s_i should be encoded in that position.

After the refinement and expansion of q-gram sets in $\mathbf{C}^P$, $\mathbf{C}^N$, and $\mathbf{C}^A$, the attack then moves to the final step, QID value reidentification. The authors have presented two reidentification approaches, one using the list of potential q-grams $\mathbf{C}^P$, and the

other using the list of not possible q-grams, $\mathbf{C}^N$. The first approach is using the list $\mathbf{C}^P$ in the same way as the reidentification process presented in the previous attack [42] (as we discussed in Section 4.2.4 and illustrated in Figure 4.1). In the second approach, the attack identifies bit positions that a q-gram can be hashed into using not possible q-gram sets for all bit positions. For instance, if a q-gram q is in sets of not possible q-grams for all bit positions except the position p_j, then the position p_j should encode q. These identified bit positions for each q-gram are then compared with BFs to assign potential QID values to each BF. The authors argued that because the precision of the sets of not possible q-grams in $\mathbf{C}^N$ are higher than the sets of potential q-grams in $\mathbf{C}^P$, the attack is able to reidentify encoded QID values more accurately using the second approach when compared to the first.

Experimental Evaluation: The proposed attack was evaluated using three databases, NCVR, UKCD, and a Michigan voter registration database. With the NCVR database the attack was able to reidentify upto 49K out of 220K encoded QID values with high precision in less than 10 minutes. Compared to the previous version of the attack, this version had improvements ranging from 4% to 95% in reidentifying encoded QID values with different parameter combinations and databases. The attack was also evaluated using BFs with multiple QID values encoded into them. However, it was less successful compared to single QID value encoding. Similar to the previous attack [42], this improved attack method also worked when the XOR-folding [181] and balancing [180] BF hardening methods were applied.

Limitations and discussion: Despite the improved reidentification accuracy, this attack method also has similar limitations as the attack method we discussed in Section 4.2.4. The attack assumes that the frequency distribution of encoded BFs in $\mathbf{D}^e$ is similar to the frequency distribution of plaintext QID values in $\mathbf{D}^p$. The accuracy of the second and third steps of the attack depends on the accuracy of assigned BF/QID value pairs, $(\mathbf{b}_i, s_i)$, in the first step. Therefore, any wrong assignments of BFs to QID values based on their frequencies will lead to wrong potential, not possible, and assigned q-gram sets for each bit position. Furthermore, according to the results presented [41], the attack is still less successful in reidentifying encoded QID values when the CLK or RBF encoding techniques are used.

4.2.6 Graph Traversal based Cryptanalysis for Bloom Filters

Mitchell et al. [139] have proposed a cryptanalysis attack based on a graph traversal mechanism on BF encoding for PPRL. A theoretical argument was presented by the authors to prove that increasing the length of q-grams, $l_q > 2$, does not increase the privacy of BF encodings in such a attack scenario.

Preliminaries and Assumptions: Unlike the previously discussed attacks, this proposed attack assumes a two-party protocol where each party follows the honest-but-curious (HBC) adversary model. Any party who is involved in the protocol can perform this attack where it is assumed that the adversary has access to the complete BF encoding information, including knowledge of any pre-processing steps, the pa-

rameter settings used for BF encoding, including the number of hash functions k, and keyed-hash message authentication code (HMAC) [115] keys if applicable. The attack also assumes that QID values are padded at both ends using two different characters before being encoded. These assumptions are considerably stronger compared to the assumptions made by the attack methods we discussed in the previous sections in this chapter.

Methodology: The attack consists of four main steps. In the first step a brute force approach is used to identify potential q-grams encoded into each BF. With two different padding characters used (one for the beginning and one for the end of a QID value), there can be $(|\Sigma| + 1)^{l_q} - 1$ possible q-grams, where $|\Sigma|$ is the number of characters in the alphabet and l_q is the length of a q-gram. Given that the adversary knows all the parameters used for BF encoding, the identification of all possible q-grams encoded in each BF can be done. A set of q-grams are assigned to each BF at the end of this step. In the second step a directed graph is constructed for each BF using the q-grams assigned to it. Q-grams with leading padding characters are labelled as *source* (starting node) and q-grams with ending padding characters are labelled as *sink* (ending node). An edge is drawn between two nodes v_i and v_j if the last $l_q - 1$ characters of v_i are the same as the first $l_q - 1$ characters of v_j. For instance, with the q-grams 'smi', 'mit', and 'ith' where $l_q = 3$, an edge between 'smi' and 'mit' is drawn because the last two characters of 'smi' is the same as the first 2 characters of 'mit'. Similarly, an edge between 'mit' and 'ith' is also drawn. In the third step all traversals from a *source* to a *sink* are considered to identify potential words that might be encoded into the corresponding BFs. These traversals can be derived using a depth first search such as simple walks and/or edge-disjoint walks [139].

In the final step a set of filters is applied to identify QID values which have high likelihood of being encoded into a BF, or to remove less likely QID values. One such filter could be encoding all potential QID values into separate BFs using the adversary's prior knowledge of secret encoding, and compare those BFs with the original BF. If there is an exact match the corresponding QID value is assigned to the original BF. Additional filters can be designed and applied based on an adversary's knowledge about the domain, such as expected word lengths, filters based on semantic knowledge, and so on [139].

Experimental Evaluation: The proposed attack was evaluated under three different experimental settings using three databases. The results have shown that the attack was able to identify 364,450 first and last names from 474,319 distinct QID values from the NCVR database accurately. All of these 76.8% reidentifications are 1-to-1 matches. When 1-to-*many* matches are considered, the attack reidentified 442,619 of first and last names, which is 93.3% of all the QID values. These results are for simple graph traversals. The authors have also shown that increasing the q-gram length l_q (i.e. going from l_q to $l_q + 1$) does not improve the privacy of the BF encoding, rather it will possibly decrease the number of candidate words obtained from a graph and therefore will make the attack more effective. In conclusion, the proposed attack was efficient and accurate in reidentifying QID values encoded in BFs.

Limitations and Discussion: Despite being efficient and accurate, this cryptanalysis attack has several limitations. First, the attack assumes that the adversary has complete knowledge of the parameter settings used for BF encoding, including the secret keys if any were used. This is highly unlikely in a real-world scenario. However, the authors argued that in a two party protocol where both parties have access to all the required information, it is possible to perform this attack. Second, the structure of a word inserted into a BF greatly impacts its recoverability. For instance, if the word has duplicate q-grams, it might lead to unnecessary edges in the graph which then lead to a large number of candidate traversals. Finally, the attack assumes that two different non-alphabetical characters are padded to the front and end of a QID value before encoding, so the start and end of the graph can be determined in the second step of the attack. However, this will not be the case in every real-world linkage scenario. If no padding is conducted, determining the source and sink nodes in a graph will be difficult and this could again lead to a large number of candidate traversals.

4.2.7 Graph Matching based Attack for Similarity Tables

Culnane et al. [46] have proposed a similarity graph based attack on a PPRL encoding technique developed by the UK Office for National Statistics (ONS) [197, 198]. Unlike the attacks we described in the previous sections, this attack was not proposed for BF encoding, but for an encoding technique named *score based matching* that uses HMAC [115] and similarity tables to identify matching record pairs. In this score based matching approach, for each QID that is being compared a similarity table is built. Such a similarity table contains pairs of hash values from candidate QID value pairs and their corresponding plaintext similarities calculated between those QID value pairs. Similarities are calculated using a Statistical Analysis System (SAS) proprietary string comparison function named SPEDIS [167]. The generated similarity tables are then used as lookup tables for identifying obtained similarities between QID values in a pair of records that need to be matched [197].

Preliminaries and Assumptions: Culnane et al. [46] have first discussed the possibility of a frequency attack using the generated similarity tables. The authors argued that by analysing the frequencies of HMAC'd hash values the most frequent QID values can potentially be reidentified. The authors have also identified that if a set of similarity values calculated for a certain QID value (by comparing it with other QID values) is unique within a similarity table, then that QID value can be reidentified. This can be done by comparing the unique set of similarities with sets of similarities calculated using a similar plaintext QID value database that is publicly available. However, these attacking scenarios have two limitations, (1) the adversary might not have access to a plaintext database with very similar QID values, and (2) potential errors of QID values in the sensitive database can make the calculated similarities differ from the similarities calculated using the plaintext database.

To overcome these limitations, Culnane et al. [46] have then presented a similarity graph matching based attack method. The attack assumes that the adversary has access to one or more similarity tables generated using a sensitive database, $\mathbf{D}^p$, and

a plaintext database, $\mathbf{D}^p$, which at least has a subset of QID values similar to $\mathbf{D}^s$. The attack also assumes that the adversary knows which QID attribute is being used to generate the similarity tables.

Methodology: The attack consists of two main steps. In the first step, it generates a similarity graph using the similarity table where nodes in the graph are the HMAC'd QID values and the edges between nodes are the similarities calculated between them. The attack also generates a set of smaller graphs using the QID values in the plaintext database and their corresponding similarities. In the second step, the attack then conducts a subgraph isomorphism to identify which plaintext graph has an isomorphism with certain parts of the larger similarity table graph.

To conduct the subgraph matching, the attack first obtains unique nodes in the plaintext graphs according to their sets of inbound and outbound edges. The attack next searches for similar unique nodes in the larger similarity table graph. If there is a matching unique node then the QID value from the plaintext node is assigned to the unique node from the similarity table graph. Then the inbound and outbound edges of the identified node is analysed to reidentify QID values of the other nodes connected by those edges based on their uniqueness. This process is continued until there is no unique nodes to be reidentified.

Experimental Evaluation: The proposed attack was evaluated using a dataset of 390,523 Australian last names. The authors have evaluated how the accuracy of the attack changes when the adversary has access to different sample sizes of the HMAC'd dataset as the plaintext dataset ranging from 90% to 10%. The results showed that the attack was able to correctly reidentify 90%, 78%, and 31% of the last names when the adversary has access to 90%, 50%, and 20% of the HMAC'd values in its plaintext form, respectively. With 10% of HMAC'd values the attack was not successful in reidentifying any HMAC'd QID values.

Limitations and discussion: The proposed attack was able to reidentify HMAC'd QID values with high precision when the adversary has access to a plaintext dataset with at least 50% of HMAC'd values. However, it is worth noting that the proposed attack method works only with score based matching PPRL technique and would not be able to apply on any other popular PPRL techniques such as BF encoding. This is because with score based matching the adversary will be able to calculate the exact similarities of pairs of QID values using the proprietary similarity function (such as SPEDIS [167]) because of its reproducibility using publicly available information. With BF encoding however, the similarities calculated between BFs are different than the similarities calculated between plaintext QID values [177] making the subgraph matching difficult.

4.3 Other Related Privacy Attacks

We now discuss three popular privacy attacks conducted on real-world databases to reidentify sensitive information. These attacks do not lie directly in the context of PPRL, but are in the context of secure data publishing. However, these attacks can be

considered as linkage attacks as we discussed in Section 3.6.1, and they are therefore worth discussing.

Sweeney [190] presented a linkage attack where an anonymised medical database from the US state of Massachusetts was linked with a voter registration list from the same state. She was able to successfully reidentify the medical record of the governor of Massachusetts at that time using this linkage attack. She also showed that 12% of voters in the voter database had a unique date of birth, and 69% of the voters had a unique value when date of birth and zipcode values are combined. Sweeney [190] identified that this uniqueness of combined QID values can be used to successfully link patient data with voter data.

Narayanan and Shmatikov [144] have conducted another successful linkage attack on a dataset released by Netflix for a user rating prediction competition, which is now commonly known as the Netflix Prize dataset. This dataset contained anonymised movie rating information of over 480,189 Netflix users. The attack was able to successfully reidentify entities from this Netflix dataset by combining those movie rating data with another dataset from the Internet Movie Database (IMDb). The authors showed that only a small amount of auxiliary information (such as rated movie names and dates when movie were rated) was enough to uniquely identify entities in the Netflix Prize dataset. The authors also argued that some potentially private information, such as one's political opinions, religious beliefs, and sexual orientation, can likely be derived from these movie ratings and that reidentification of records will therefore lead to a breach of privacy [144].

A privacy attack conducted by Culnane et al. [47] on a supposedly anonymised Australian medical billing record database was able to recognise unique patterns such as childbirth patterns, unique total payments, and unique surgeries and injuries. The medical database was published by the Australian Department of Health in 2016 under the Australian government's policy of open government data. The authors have shown that people in this database can be identified by their names by linking the identified unique patterns with publicly available information [47].

4.4 Summary

In this chapter we conducted an analysis of existing attack methods proposed on different PPRL techniques. We summarised the methodologies used in these attacks and discussed their requirements and limitations. This analysis helped us to identify potential research directions with respect to privacy attacks on PPRL that have not been explored so far. Following those identified research avenues, we propose three novel attack methods on different PPRL techniques in Chapters 6 to 8.

Chapter 5

A Vulnerability Assessment Framework for Privacy-Preserving Record Linkage

All existing attack methods on privacy-preserving record linkage (PPRL), including the attacks we propose in Chapters 6 to 8, exploit different vulnerabilities in PPRL techniques and encoded databases in order to reidentify encoded quasi-identifying attribute values. Therefore, it is important that database owners can identify those vulnerabilities associated with their databases and choice of PPRL technique before those databases are being used in a PPRL project. In this chapter we present a novel framework for assessing the vulnerabilities of sensitive databases to be linked, and of PPRL encoding techniques.

In Section 5.1 we first highlight the importance of assessing vulnerabilities in PPRL and in Section 5.2 we introduce the vulnerability framework and outline our contributions. In Section 5.3 we describe the worst-case scenario of how a PPRL system can be attacked, and in Section 5.4 we then provide an overview of the proposed vulnerability framework. In Sections 5.5 and 5.6 we next describe the process of quantifying the vulnerabilities of plaintext and encoded databases, respectively. In Section 5.7 we discuss the applicability of different vulnerabilities to different plaintext values and PPRL encoding techniques, and in Section 5.8 we analyse existing attacks on PPRL based on the identified vulnerabilities. In Section 5.9, using an experimental evaluation, we illustrate and discuss how this framework can be used to assess the vulnerabilities of plaintext and encoded databases. We conclude our work with a brief summary in Section 5.10.

5.1 Introduction

As we discussed in Chapter 1, the use of personal information in a sensitive database for record linkage (RL) often leads to ethical and legal concerns associated with privacy [29, 209]. Various privacy-preserving record linkage (PPRL) [87] techniques have been proposed to address these concerns by linking sensitive databases without

compromising the privacy of the entities whose records are being linked. However, as we discussed in Chapter 4, certain PPRL techniques, including popular Bloom filter (BF) encoding [177], have shown to be susceptible to privacy attacks under certain assumptions and specific parameter selections [41, 46, 121, 139].

A systematic vulnerability assessment is an important step towards understanding the risks associated with a sensitive database or an encoding technique that is to be used in a PPRL project. Identifying and quantifying vulnerabilities that an adversary might exploit will help database owners (DOs) to take the necessary precautions to ensure their sensitive databases will have no (or only acceptable) reidentification risks against a possible privacy attack. However, existing methods [63, 64, 200] to measure the privacy risks associated with sensitive databases and encoding techniques in the context of PPRL are ad-hoc and do not formally assess the vulnerabilities of sensitive data. The risk of disclosure of sensitive data is however, widely discussed in the areas of data publishing [77] and statistical disclosure control [59, 71, 193]. Most existing disclosure risk measures focus on the concept of *uniqueness* [71, 192, 200]. A record in a given database can be defined as unique if that record shares its QID values with only up to n other records in the same database, with $0 \leq n < k$ (with regard to the k-anonymity model [191, 217]) [59]. Based on this notion of uniqueness the disclosure risk of records in a database can be measured [191, 195]. We refer the interested reader to Appendix B for more details on disclosure risk measures proposed in the area of statistical disclosure control.

Two PPRL based measures have been proposed to quantify disclosure risks associated with sensitive data. Durham et al. [63] and Karakasidis and Verykios [104] proposed to use relative information gain (RIG) as a measure of privacy for an encoded database. Vatsalan et al. [204] have then proposed to use probability of suspicion (PS) to calculate the identity disclosure risk in an encoded database. We discussed more details about these two measures in Section 2.6.3.

Even though these existing disclosure risk measures for PPRL are based on the uniqueness of the values in a sensitive database, the notion of uniqueness is explored only in one dimension. That is, if a record or a quasi-identifying (QID) attribute value in a database has a unique combination of values across that database, an adversary could potentially reidentify that record or QID value. However, as we discuss in the following, the uniqueness of values can be explored across multiple aspects.

5.2 Contribution

In this chapter we propose a vulnerability assessment framework that allows a DO to quantify how vulnerable a sensitive database is, as well as how vulnerable an encoded version of that database will be after a certain PPRL encoding technique has being applied. We analyse the vulnerability of values in a database using five aspects: the frequency of occurrences of a value, the length of a value, the co-occurrence of two or more values, the similarity between a pair of values, and the similarity neighbourhood of a value. Such a framework will allow DOs to establish

Table 5.1: Specific notation used throughout this chapter. For common notation used throughout the thesis, refer to the list of notation at the beginning of the thesis.

$\mathbf{T}$, $\mathbf{S}$	Lists of tokens and token substrings in $\mathbf{D}^s$
$\mathbf{V}$	Set of all the unique plaintext values in $\mathbf{D}^s$
$\mathbf{H}$	Set of all the unique encodings in $\mathbf{D}^e$
$\mathbf{p}$	Set of parameter values used by the encoding function $encode()$
$\mathbf{m}$	Set of microdata where $\mathbf{m} \in \mathbf{M}$
$q \in \mathbf{q}$	A plaintext QID value in a set of QID values where $\mathbf{q} \in \mathbf{Q}^s$
$e \in \mathbf{e}$	An encoding in a set of encodings where $\mathbf{e} \in \mathbf{E}$
$t \in \mathbf{t}$	A token in a set of tokens where $\mathbf{t} \in \mathbf{T}$
$s \in \mathbf{s}$	A token substring in a set of token substrings where $\mathbf{s} \in \mathbf{S}$
v	Common notation used to represent all three types of values q, t, and s
$\mathcal{E}$	Real-world entities represented by the records in $\mathbf{D}^s$
ϵ	An entity where $\epsilon \in \mathcal{E}$
c	Secret key used in the encoding process
ε	Maximum difference limit of two values
k	Maximum limit of the number of similar values a set should contain

the necessary measures to either modify their databases (for example adding random noise [67]), select an alternative PPRL technique that provides enough privacy guarantees against any possible reidentification of an entity, or try to strengthen the privacy guarantees of an existing PPRL technique. In an experimental evaluation we assess the vulnerabilities of two databases using five different PPRL encoding techniques. This evaluation shows that our proposed framework can be used in real-world linkage situations to assess the vulnerabilities associated with sensitive databases to be linked, as well as with PPRL encoding techniques.

In the following we first describe the worst-case attack scenario that can occur in a given PPRL situation and then discuss the proposed vulnerability framework in detail. Specific notation used throughout the chapter are shown in Table 5.1.

5.3 Worst-case Attack Scenario

Assume a sensitive database $\mathbf{D}^s = (\mathbf{Q}^s, \mathbf{M})$, where $\mathbf{Q}^s$ is the list of plaintext QID values and $\mathbf{M}$ is the list of plaintext microdata. The encoded database of $\mathbf{D}^s$ is defined as $\mathbf{D}^e = (\mathbf{E}, \mathbf{M})$, where $\mathbf{E}$ is the list of encoded QID values. The worst-case scenario for a privacy attack in PPRL is where an adversary has access to all the plaintext QID values, $\mathbf{Q}^s$, from the sensitive database $\mathbf{D}^s$, the encoded QID values, $\mathbf{E}$, and the associated plaintext microdata, $\mathbf{M}$, from the encoded database $\mathbf{D}^e$. The adversary also knows the encoding function, $encode()$, and its parameter settings, $\mathbf{p}$, used. The only information the adversary does not possess is the secret key, c, used in the encoding function $encode()$. The adversary also does not have access to the full sensitive database $\mathbf{D}^s = (\mathbf{Q}^s, \mathbf{M})$, and therefore she does not know which set of

microdata values $\mathbf{m}_i \in \mathbf{M}$ refers to which set of QID values $\mathbf{q}_i \in \mathbf{Q}^s$, and therefore belongs to which entity $e_i \in \mathcal{E}$.

In the context of cryptography, such an attack scenario is known as a *known plaintext* attack [111], where an adversary aims to assign a set of plaintext QID values $\mathbf{q}_i$ to a set of encodings $\mathbf{e}_j$ so that the identity of the person with the corresponding set of sensitive microdata $\mathbf{m}_i$ can be identified. By doing so, the adversary is able to learn sensitive information (such as certain illnesses or confidential financial details) of a given entity e_i that is represented by a record $r_i \in \mathbf{D}^s$. Based on the information the adversary has access to, such as frequency distributions of plaintext and encoded QID values, she can try to assign sets of encodings $\mathbf{e}_j \in \mathbf{E}$ to an entity $e_i \in \mathcal{E}$ such that the encodings for QID values $\mathbf{q}_i$ of that entity e_i are $encode(\mathbf{q}_i, \mathbf{p}, c) = \mathbf{e}_j$. An attack is successful if the adversary is able to obtain one or more correct such assignments between $\mathbf{e}_j \in \mathbf{E}$ and $e_i \in \mathcal{E}$.

In practice, this worst-case attack scenario is unlikely to happen as it would require the adversary to have access to almost all information used in the PPRL encoding process (except the secret key, c). This would only be possible if the adversary is an inside attacker. In reality, the adversary will likely have access to less information used in the encoding process, or have access to a plaintext database that contains QID values and/or records that cover a somewhat different population than the one covered by $\mathbf{Q}^s$.

However, the adversary can potentially identify the encoding function $encode()$ that is used to encode the QID values in the sensitive database $\mathbf{D}^s$ by analysing the encodings in $\mathbf{E}$. We discuss such practical attack scenarios in detail in Section 3.5. Despite that this worst-case scenario is unlikely to occur in real-world situations, it allows us to assess the vulnerabilities of the QID values in $\mathbf{Q}^s$, as well as the encodings in $\mathbf{E}$, as we discuss in the following section.

5.4 Overview of the Vulnerability Assessment Framework

In the following we assume that each record $r_i \in \mathbf{D}^s$ has a set of QID values $\mathbf{q}_i \in \mathbf{Q}^s$, such as first and last name, street address, and date of birth. Each of these QID values can contain one or several *tokens* (words and numbers separated by a whitespace character). Data cleaning and pre-processing might have removed hyphens and dashes, resulting for example in a person having several first name tokens (like 'Hans Dieter' or 'Jean Pierre'). While addresses generally contain several tokens (such as '42 Miller Street'), values in most QIDs, such as names, city, and postcode, commonly contain only one token. We denote a single token as t, and the set of tokens that represent the record $r_i \in \mathbf{D}^s$ as $\mathbf{t}_i$.

Tokens $t \in \mathbf{t}_i$ can then be converted into a set of token *substrings*, $\mathbf{s}_i$, which can be used in the encoding function $encode()$ to encode the record, r_i. A token substring $s \in \mathbf{s}_i$ is a segment of a token, potentially with a specific length. A common type of token substring used in PPRL encoding techniques (such as BF encoding [177]) is a q-gram [38]. Similar to $\mathbf{Q}^s$, we denote $\mathbf{T}$ as the list of all token sets in $\mathbf{D}^s$, and $\mathbf{S}$ as the

Table 5.2: An example record r_i in the sensitive database $\mathbf{D}^s$ and how that record can be converted into a set of QID values ($\mathbf{q}_i$), a set of tokens ($\mathbf{t}_i$), and a set of token substrings ($\mathbf{s}_i$). Here q-grams (character substrings of length $l_q = 2$) are used as the set of token substrings $\mathbf{s}_i$. The record r_i contains first name, last name, street address, and city values of an entity $e_i \in \mathcal{E}$. We also show the encoded values obtained for the record r_i using four encoding techniques, Bloom filters (BF) [177], tabulation min-hashing (TMH) [186], multiple match-keys (MMK) [163], and two-step hashing (2SH) [158]. Note that the set of microdata values $\mathbf{m}_i$ is also a part of the record r_i, but these are not shown here as they are not being encoded.

	FirstName	*LastName*	*StreetAddress*	*City*
Record (r_i)	Jean Pierre	Miller	42 Miller Street	Chapel Hill
QID values ($\mathbf{q}_i$)	Jean Pierre, Miller, 42 Miller Street, Chapel Hill			
Tokens ($\mathbf{t}_i$)	Jean, Pierre, Miller, 42, Street, Chapel, Hill			
Token substrings ($\mathbf{s}_i$)	an, ap, ch, ea, ee, el, er, et, ha, hi, ie, il, je, le, ll, mi, pe, pi, re, rr, st, tr, 42			
BF Encoding	00101110011011111010001111101111111111100101011010			
TMH Encoding	01000101010110111100111011001111110111100110110001			
MMK Encoding	i+sL9RtXd4Jb, avxxlTxIoLx3, Lr1dnWLGM/K8, fOyHBWsUEYr6			
2SH Encoding	128, 2, 8, 143, 16, 146, 148, 26, 155, 28, 34, 37, 166, 168, 45, 136, 178, 56, 59, 185, 61, 195, 70, 162, 79, 89, 95, 100, 108, 119, 122, 127			

list of all token substring sets in $\mathbf{D}^s$, where one token set $\mathbf{t}_i \in \mathbf{T}$ or token substring set $\mathbf{s}_i \in \mathbf{S}$ refers to a record $r_i \in \mathbf{D}^s$.

In Table 5.2 we show an example record $r_i \in \mathbf{D}^s$ and how this record is separated into QID values $\mathbf{q}_i$, tokens $\mathbf{t}_i$, and token substrings $\mathbf{s}_i$. As can be seen, a record with one or more QID values can contain the same tokens. For instance, the token 'Miller' occurs in both the last name and street address QID values. Similarly, different tokens can contain the same token substrings. For instance, the q-gram (token substring) 're' occurs in both the tokens 'Pierre' and 'Street', as shown in Table 5.2.

Depending on the encoding technique used, the encoding function $encode()$ uses either QID values, tokens, or token substrings, as input to generate the set of encodings $\mathbf{e}_i$ to represent the record r_i. For instance, BF encoding [177] uses q-grams (token substrings) in its encoding function, while multiple match-key (MMK) encoding [163] uses QID value combinations in its encoding function. We assume that for each record r_i, one of its sets of QID values, $\mathbf{q}_i$, tokens, $\mathbf{t}_i$, or token substrings, $\mathbf{s}_i$, is being encoded in the PPRL process using the encoding function $encode()$. In Table 5.2 we show the output encodings for four different PPRL encoding techniques. For instance, as can be seen, BF encoding outputs a single bit vector (if cryptographic long-term key (CLK) encoding is used), whereas MMK encoding outputs a set of hash values.

In Figure 5.1 we show the general idea behind an attack on sensitive data in the context of PPRL. As illustrated, a unique entity $e_i \in \mathcal{E}$ in a sensitive database

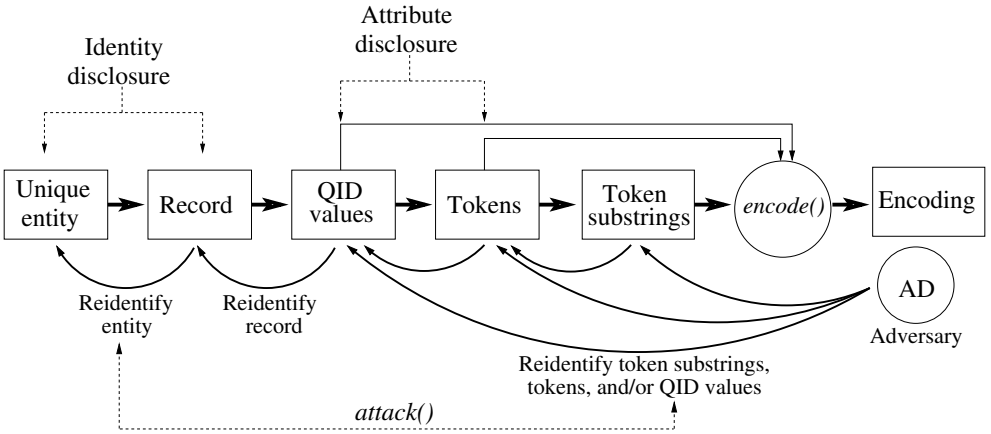


Figure 5.1: The overall idea behind a privacy attack on encoded data in the context of PPRL illustrated for a single record in the sensitive database $\mathbf{D}^s$. QID values, tokens, or token substrings, can be used as input to the encoding function $encode()$ which generates encodings for each record $r_i \in \mathbf{D}^s$. We illustrate the attack function, $attack()$, by left pointing arrows. By analysing the generated encodings, an adversary can try to reidentify token substrings, tokens, or QID values, encoded into different encodings. If the adversary is successful in reidentifying token substrings, she can next try to reidentify tokens using those identified token substrings, and then try to reidentify QID values from the identified tokens. This process continues until the adversary is able to reidentify the entities whose records (containing sensitive information) have been encoded. It is also possible for the adversary to reidentify tokens or QID values (without reidentifying token substrings or tokens) directly from the encoded values as we illustrate in the figure.

can be represented by a set of QID values $\mathbf{q}_i$, a set of tokens, $\mathbf{t}_i$, or a set of token substrings $\mathbf{s}_i$. In an attack, the adversary aims to reverse-engineer the encoding process, starting from the list of encoding sets, $\mathbf{E}$, to first reidentify a set of one or more token substrings, $\mathbf{s}_i \in \mathbf{S}$, tokens, $\mathbf{t}_i \in \mathbf{T}$, or QID values, $\mathbf{q}_i \in \mathbf{Q}^s$, that have been used for the encoding of the record r_i. Each encoding set $\mathbf{e}_j \in \mathbf{E}$ will then be assigned to zero, one, or more such reidentified token substrings, tokens, or QID values. The values (token substrings, tokens, or QID values) that are assigned to encodings $\mathbf{e}_j \in \mathbf{E}$ can potentially lead to *attribute disclosure* in certain records $r_i \in \mathbf{D}^s$. For instance, one or more reidentified token substrings might provide sufficient information to reidentify tokens, and some of these reidentified tokens might lead to the reidentification of QID values. From these reidentified QID values, *identity disclosure* of entities $\epsilon_i \in \mathcal{E}$ could be possible depending upon the uniqueness of the values $q \in \mathbf{q}_i$ as they occur in $\mathbf{Q}^s$. A successful identity disclosure occurs when the adversary can assign a set of encodings $\mathbf{e}_j \in \mathbf{E}$ to an entity $\epsilon_i \in \mathcal{E}$ where for the corresponding QID values $\mathbf{q}_i$ of that entity ϵ_i it holds that $encode(\mathbf{q}_i, \mathbf{p}, c) = \mathbf{e}_j$.

This leads us to the concept of vulnerability. For attribute and/or identity disclosure to be successful, certain QID values $q \in \mathbf{Q}^s$, tokens $t \in \mathbf{T}$, token substrings $s \in \mathbf{S}$, or encodings $e \in \mathbf{e}_j$, where $\mathbf{e}_j \in \mathbf{E}$, need to be vulnerable to allow an adversary

to successfully conduct her attack. Since a vulnerable plaintext value can refer to either a QID value $q \in \mathbf{Q}^s$, a token $t \in \mathbf{T}$, a token substring $s \in \mathbf{S}$, or to all of them, without loss of generality, we use the notation v to represent all three types of values q, t, and s. We define a set $\mathbf{V}$ which consists of all the unique values v in a database which are to be encoded. Similarly, we define a set $\mathbf{H}$ which consists of all the unique encodings $e \in \mathbf{e}_i$ where $\mathbf{e}_i \in \mathbf{E}$. Each value $v \in \mathbf{V}$ or encoding $e \in \mathbf{H}$ has a list of characteristics assigned to it, such as its frequency, co-occurrence, or length as we discuss in the following section.

A value $v \in \mathbf{V}$ and an encoding $e \in \mathbf{H}$ becomes vulnerable, and an encoding in $e \in \mathbf{H}$ becomes reidentifiable based on two conditions. Following the concept of k-anonymity [191, 217] (as we discuss in Appendix B), we next define these two conditions that are associated with the vulnerability and the reidentifiability. We use two privacy parameters, ε and k, where ε defines the maximum difference limit of two values, for those to be considered as non-distinguishable from each other, and k defines the maximum limit of the number of similar values a set should contain for those values to be considered as vulnerable.

Definition 5.4.1. Plaintext (ε, k)-*Vulnerability*
For a plaintext value $v_i \in \mathbf{V}$ and the set $\mathbf{v}_i = \{v_j : func(v_i, v_j) \leq \varepsilon, v_i \neq v_j\}$ of other plaintext values $v_j \in \mathbf{V}$, with a tolerance $\varepsilon \geq 0$, we define v_i as (ε, k)-vulnerable in the database $\mathbf{D}^s$ with regard to the function $func()$ if $0 \leq |\mathbf{v}_i| < k$.

For a plaintext value $v \in \mathbf{V}$ to become vulnerable, it needs to satisfy the condition in the Definition 5.4.1 with regard to the selected values for ε and k. Therefore, we define a vulnerable plaintext value $v \in \mathbf{V}$ to be one that is unique or rare in the corresponding list $\mathbf{Q}^s$, $\mathbf{T}$, or $\mathbf{S}$ with regard to some characteristics as we describe below. Similarly, encodings $e \in \mathbf{H}$ also become vulnerable, as well as reidentifiable, based on the following conditions.

Definition 5.4.2. Encoding (ε, k)-*Vulnerability*
For an encoding $e_i \in \mathbf{H}$ and the set $\mathbf{e}_i = \{e_j : func(e_i, e_j) \leq \varepsilon, e_i \neq e_j\}$ of other encodings $e_j \in \mathbf{H}$ with a tolerance $\varepsilon \geq 0$, we define e_i as (ε, k)-vulnerable in the database $\mathbf{D}^e$ with regard to the function $func()$ if $0 \leq |\mathbf{e}_i| < k$.

Definition 5.4.3. (ε, k)-*Assignability*
For an encoding $e_i \in \mathbf{H}$, that is (ε, k)-vulnerable within the set $\mathbf{H}$, we define a set $\mathbf{m}_i = \{v_a : v_a \in \mathbf{V}, func(e_i, v_a) \leq \varepsilon\}$ of vulnerable plaintext values $v_a \in \mathbf{V}$ that can be assigned to the vulnerable encoding e_i, based on a function $func()$ and a tolerance ε. If $1 \leq |\mathbf{m}_i| < k$, with k and $\varepsilon \geq 0$ being privacy parameters, then we define the pair of encoding and plaintext value, (e_i, v_a), as (ε, k)-assignable.

For an encoding $e \in \mathbf{H}$ to be vulnerable, it needs to satisfy the condition defined in the Definition 5.4.2 where the encoding should be unique or rare in the corresponding list $\mathbf{E}$. However, for the encoding $e \in \mathbf{H}$ to becomes reidentifiable, it also needs to satisfy the condition defined in the Definition 5.4.3, as we further discuss in Section 5.6. Therefore, a reidentifiable encoding $e \in \mathbf{H}$ should not only be vulnerable in list $\mathbf{E}$ (i.e. the encoding is unique within the set of encodings $\mathbf{H}$), but should

also be assignable to a unique or rare plaintext value $v \in \mathbf{V}$ (or a small number of plaintext values) with regard to some characteristics. A vulnerable plaintext value or a reidentifiable encoding can lead to attribute disclosure which potentially leads to identity disclosure of the corresponding entity as we discussed above.

In the above definitions the function $func()$ can be any measure that calculates the difference between the two plaintext values v_i and v_j, two encodings e_i and e_j, or an encoding and plaintext value e_i and v_a, based on some characteristics as we discuss in the following sections. Furthermore, this definition can also be extended to characterise the vulnerability of a set of plaintext values $\mathbf{v}_a$ or a set of encodings $\mathbf{e}_a$, where $|\mathbf{v}_a|, |\mathbf{e}_a| \geq 2$, depending on the characteristic of values that is being analysed.

In the following we discuss five different characteristics (that corresponds to the function $func()$) that can be used to assess the vulnerabilities of plaintext values (QID values, tokens, and token substrings) and encodings and how the general definition of vulnerability differs for each of those characteristics in detail.

5.5 Assessing Plaintext Vulnerabilities

We first discuss the vulnerabilities in plaintext QID values, $q \in \mathbf{Q}^s$, tokens, $t \in \mathbf{T}$, and token substrings $s \in \mathbf{S}$, because this will allow us to analyse the vulnerabilities of the sensitive plaintext database $\mathbf{D}^s$ before being encoded using a suitable PPRL technique. Understanding how vulnerable certain QID values, tokens, or token substrings in a sensitive database are will allow a DO to make a decision to select a suitable PPRL encoding technique, modify the database (for example, removing records that are too vulnerable to be linked), or abandon a linkage project altogether.

When considering values that occur in records (QID values, tokens, and token substrings) in a sensitive database, the vulnerabilities that occur can be classified into two categories: vulnerabilities of a single value and vulnerabilities of a pair of values as we describe next.

5.5.1 Vulnerability of a Single Value

Three different types of vulnerabilities can occur under this category that potentially can be exploited by an adversary: the frequency of a value, the length of a value, and the similarity neighbourhood of a value.

- **Frequency Vulnerability**: Frequency vulnerability is the vulnerability of a value with regard to how uniquely or rarely it occurs in records in the sensitive database $\mathbf{D}^s$. We assume a function $freq(v_i)$ which returns the frequency of v_i as the number of records in $\mathbf{D}^s$ that contain v_i. The value $v_i \in \mathbf{V}$ with a frequency of $freq(v_i)$ is frequency vulnerable if $freq(v_i)$ is different from the frequencies $freq(v_j)$ of other values $v_j \in \mathbf{V}$. Formally, for v_i and the set $\mathbf{f}_i = \{v_j : v_j \in \mathbf{V}, v_j \neq v_i, |freq(v_i) - freq(v_j)| \leq \varepsilon\}$ of values that each has a frequency similar to v_j based on a tolerance ε, with k and $\varepsilon \geq 0$ being two pri-

vacy parameters, then the value v_i is (ε, k)-frequency vulnerable in the sensitive database $\mathbf{D}^s$ if $0 \leq |\mathbf{f}_i| < k$ for a given ε.

Frequency vulnerability can occur when the frequency of a value is uniquely identifiable. For example, the first and second most frequent American last names are 'Smith' and 'Johnson', respectively[1]. If a sensitive database $\mathbf{D}^s$ contains these two last names with the frequencies $freq(\text{Smith}) = 400$ and $freq(\text{Johnson}) = 300$, then the value 'Smith' becomes frequency vulnerable under the privacy parameter settings $\varepsilon < 100$ and $k > 0$. This is because for these parameter values the frequency of 'Smith' is different from the frequency of 'Johnson'. On the other hand if the sensitive database $\mathbf{D}^s$ contains the three least frequent last name values 'Samarasinghe', 'Arachchige', and 'Widana' with the frequencies $freq(\text{Samarasinghe}) = 5$, $freq(\text{Arachchige}) = 6$, and $freq(\text{Widana}) = 10$ then 'Samarasinghe' becomes frequency vulnerable under the privacy parameter settings $\varepsilon < 5$ and $k > 1$ while 'Arachchige' becomes frequency vulnerable under $\varepsilon < 4$ and $k > 1$.

- **Length Vulnerability**: Length vulnerability is the vulnerability of a value with regard to how unique its length is compared to the lengths of other values of the same QID attribute in the sensitive database $\mathbf{D}^s$. We assume a function $len(v_i)$ which returns the length of a value (assumed to be a string) as the number of characters it contains. A given value $v_i \in \mathbf{V}$ with a length of $len(v_i)$ is length vulnerable if the length $len(v_i)$ is different from the lengths $len(v_j)$ of other values in $\mathbf{V}$. Formally, for v_i and the set $\mathbf{l}_i = \{v_j : v_j \in \mathbf{V}, v_j \neq v_i, |len(v_i) - len(v_j)| \leq \varepsilon\}$ of values that each has a length similar to v_j based on a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then v_i is (ε, k)-length vulnerable in $\mathbf{D}^s$ if $0 \leq |\mathbf{l}_i| < k$ for a given ε.

  Similar to frequency vulnerability, length vulnerability can occur in any situation where the length of a value is uniquely identifiable. For instance, one of the longest last names recorded in modern databases is 'Wolfeschlegelsteinhausenbergerdorff'[2] with the length of 35 characters. If the sensitive database $\mathbf{D}^s$ contains this last name and a second longest last name is 'Kellermann' with the length of 10 characters, then the value 'Wolfeschlegelsteinhausenbergerdorff' becomes frequency vulnerable under the privacy parameter settings $\varepsilon < 25$ and $k > 0$.

- **Similarity Neighbourhood Vulnerability**: Similarity neighbourhood vulnerability is defined with regard to how unique the neighbourhood of a value in a similarity graph (based on a set of extracted features) is compared to the neighbourhoods of other values in the same similarity graph. Here the similarity graph is build using the pairwise similarities between records in the sensitive database $\mathbf{D}^s$ [46, 214]. We define the neighbourhood of a value v_i as $\mathbf{v}_i = \{v_j : v_j \in \mathbf{V}, v_j \neq v_i, sim(v_i, v_j) \geq s_t\}$, where $\mathbf{v}_i$ is a set of other values

[1] https://namecensus.com/most_common_surnames.htm
[2] https://en.wikipedia.org/wiki/Hubert_Blaine_Wolfeschlegelsteinhausenbergerdorff_Sr.

with each value pair (v_i, v_j) having a similarity $sim(v_i, v_j)$ of at least a user defined similarity threshold, s_t. We next assume a function $feat(\mathbf{v}_i)$ which returns a set of features for the value v_i calculated using its neighbourhood $\mathbf{v}_i$ [214]. The features can include the degree (number of neighbours) of v_i, minimum, maximum, and average pairwise similarities between v_i and its neighbours $v_j \in \mathbf{v}_i$, and egonet and centrality of the value v_i [3]. The value v_i is similarity neighbourhood vulnerable if its neighbourhood $\mathbf{v}_i$ is different from the neighbourhood sets $\mathbf{v}_p$ of other values in $\mathbf{V}$ based on the distance between their corresponding feature sets $feat(\mathbf{v}_i)$ and $feat(\mathbf{v}_p)$.

Formally, for a value v_i, a set $\mathbf{n}_i = \{v_p : v_p \in \mathbf{V}, v_p \neq v_i, dist(feat(\mathbf{v}_i), feat(\mathbf{v}_p)) \leq \varepsilon\}$ can be defined where $dist()$ is the function to measure the distance between two vectors. The set $\mathbf{n}_i$ consists of values where each has a neighbourhood that is similar to the neighbourhood of the value v_i based on a tolerance ε. If $0 \leq |\mathbf{n}_i| < k$, with k and $\varepsilon \geq 0$ being two privacy parameters, then v_i is (ε, k)-neighbourhood vulnerable in the sensitive database $\mathbf{D}^s$. Here the distance function $dist()$ can be any function which calculates the distance between two real-valued vectors such as Euclidean or Cosine distance [38].

5.5.2 Vulnerability of a Pair of Values

Two different types of vulnerabilities can occur under this category that potentially can be exploited by an adversary: the co-occurrence of two (or more) values and the similarity between a pair of values.

- **Co-occurrence Vulnerability**: The vulnerability of a set of two or more values with regard to how unique the co-occurrence frequency of that set of values is in the sensitive database $\mathbf{D}^s$. We assume a function $freq(\mathbf{v}_i)$ which returns the frequency of the set of values $\mathbf{v}_i$ that co-occur with each other as the number of records in $\mathbf{D}^s$ that contain all $v_i \in \mathbf{v}_i$, where $|\mathbf{v}_i| \geq 2$. A given set of values $\mathbf{v}_i \in \mathbf{V}$ that co-occur with each other with a frequency of $freq(\mathbf{v}_i)$ is co-occurrence vulnerable if the frequency $freq(\mathbf{v}_i)$ is different from the frequencies $freq(\mathbf{v}_j)$ of other sets of co-occurring values $\mathbf{v}_j \in \mathbf{V}$. Formally, for a set of values $\mathbf{v}_i$, the set $\mathbf{F}_i = \{\mathbf{v}_j : \mathbf{v}_j \in \mathbf{V}, \mathbf{v}_i \neq \mathbf{v}_j, |freq(\mathbf{v}_i) - freq(\mathbf{v}_j)| \leq \varepsilon\}$ of sets of values that each has a co-occurring frequency similar to $freq(\mathbf{v}_i)$ based on a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the set $\mathbf{v}_i$ is (ε, k)-co-occurrence vulnerable in $\mathbf{D}^s$ if $0 \leq |\mathbf{F}_i| < k$.

  Co-occurrence vulnerability can happen even if the co-occurring values are not individually frequency vulnerable. For instance, let us assume two first name tokens 'Hans' and 'Dieter', and one last name token 'Schmidt' with the individual frequencies $freq(\text{Hans}) = 110$, $freq(\text{Dieter}) = 100$, and $freq(\text{Schmidt}) = 90$, where all these values are not (ε, k)-frequency vulnerable with $\varepsilon = 0$ and $k = 3$ if we assume that for each of these tokens there are more than two other tokens in the sensitive database $\mathbf{D}^s$ with the same frequency. However, if the co-occurring frequency $freq(\text{Hans, Dieter, Schmidt}) = 10$ is different from the

frequencies of other co-occurring sets of values because there are no more than two other sets of tokens that co-occur with the same frequency in $\mathbf{D}^s$, then the token tuple (Hans, Dieter, Schmidt) will be co-occurrence vulnerable under the privacy parameter settings $\varepsilon = 0$ and $k = 3$.

Note that the co-occurrence vulnerability is independent from the frequency vulnerability discussed above. If a set of values co-occur in records with a unique frequency then this does not imply that each value in that set is frequency vulnerable. Similarly, if any value in a co-occurring set of values is individually frequency vulnerable, that does not imply that the set of values is co-occurrence vulnerable. For example, let us assume two first name tokens 'Jean' and 'Pierre' with the individual frequencies $freq(\text{Jean}) = 400$ and $freq(\text{Pierre}) = 300$, are frequency vulnerable because their frequencies are unique in the sensitive database. However, the co-occurring frequency of the token pair $freq(\text{Jean, Pierre}) = 20$ might not be unique because there can be other pairs of first name tokens with the same or similar frequencies.

- **Similarity Vulnerability**: Similarity vulnerability is the vulnerability of a pair of values with regard to how unique the similarity between those two values is compared to the pairwise similarities calculated between other values in the sensitive database $\mathbf{D}^s$. We assume a function $sim(v_i, v_j)$ which returns the similarity between a value pair (v_i, v_j). A given value pair (v_i, v_j), where $v_i \in \mathbf{V}$ and $v_j \in \mathbf{V}$, is similarity vulnerable if the similarity $sim(v_i, v_j)$ is different from the similarities $sim(v_a, v_b)$ of all other value pairs (v_a, v_b) where $v_a \in \mathbf{V}$ and $v_b \in \mathbf{V}$. Formally, for a value pair (v_i, v_j) that has a similarity $sim(v_i, v_j)$ and the set $\mathbf{s}_{ij} = \{(v_a, v_b) : v_a \in \mathbf{V}, v_b \in \mathbf{V}, (v_a, v_b) \neq (v_i, v_j), |sim(v_i, v_j) - sim(v_a, v_b)| \leq \varepsilon\}$ of value pairs that each has a similarity similar to $sim(v_i, v_j)$ based on a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the value pair (v_i, v_j) is (ε, k)-similarity vulnerable in the sensitive database $\mathbf{D}^s$ if $0 \leq |\mathbf{s}_{ij}| < k$.

  Similarity vulnerability can occur if the similarity of a pair of values is uniquely identifiable. For instance, let us assume two value pairs (Miller, Mills) and (Smith, Smyth) that have Levenshtein edit distance similarities [129] of 0.67 and 0.8, respectively. The value pair (Miller, Mills) becomes similarity vulnerable under the privacy parameter settings $\varepsilon < 0.1$ and $k > 0$ when no other pair of values in the sensitive database $\mathbf{D}^s$ has a similarity between 0.57 and 0.77.

5.6 Assessing Encoded Vulnerabilities

We now discuss the vulnerabilities of encodings $e \in \mathbf{e}_i$ where $\mathbf{e}_i \in \mathbf{E}$, which, unlike plaintext vulnerabilities, will depend on the actual PPRL encoding technique used. As illustrated in Figure 5.1, for a record $r_i \in \mathbf{D}^s$, a set of encodings $\mathbf{e}_i$ will be generated using the function $encode()$, where $|\mathbf{e}_i| \geq 1$. Depending on the encoding function used, each individual encoding $e \in \mathbf{e}_i$ will be different as we illustrated in Table 5.2 on page 85. Understanding the vulnerabilities associated with each encoding tech-

nique will allow a DO to select an appropriate PPRL encoding technique according to their privacy requirements (in terms of vulnerability assessment results).

In Section 5.4 we discussed two conditions that need to be satisfied in order for an encoding $e \in \mathbf{H}$ to be reidentifiable in an attack: the encoding should be (1) vulnerable within the set $\mathbf{H}$, and (2) assignable to a unique (or a small number of) plaintext values v in the set $\mathbf{V}$. It is important to note that being only vulnerable within the set of encodings $\mathbf{H}$ does not necessarily make an encoding reidentifiable. For instance, assume an encoding e_i with a frequency $f(e_i) = 150$ is unique in terms of frequencies within $\mathbf{H}$. However, if there is no token substring, token, or QID value in the set of plaintext values $\mathbf{V}$ that has a frequency similar to the frequency of e_i (based on a defined tolerance ε), then the encoding e_i cannot be assigned to a plaintext value and therefore the plaintext value(s) encoded in e_i cannot be reidentified. In contrast, if there are at least k (for a given value of k) QID values, tokens, or token substrings in $\mathbf{V}$ that have frequencies similar to the frequency of e_i, then again the encoding e_i cannot be assigned to a plaintext value with high confidence. Note that the defined values for ε and k for encoding vulnerabilities can be different from the values defined for plaintext vulnerabilities. Therefore, in an attack, a reidentification occurs only when a vulnerable encoding $e_i \in \mathbf{H}$ can be correctly assigned to a vulnerable plaintext value $v_i \in \mathbf{V}$ that refers to the same entity ϵ_i.

Similar to plaintext vulnerabilities, as we discussed in Section 5.5, when considering encodings in a encoded database, the vulnerabilities that occur can be classified into two categories: vulnerabilities of a single encoded value and vulnerabilities of a pair of encoded values. Note that from here onwards $\mathbf{e}_x$ and/or $\mathbf{e}_y$ are used to represent a set of encodings in $\mathbf{H}$ which is different from the set $\mathbf{e}_i \in \mathbf{E}$ we used to represent the set of all encodings generated using the encoding function $encode()$ for a single record $r_i \in \mathbf{D}^s$.

5.6.1 Vulnerability and Assignability of a Single Encoding

Three types of characteristics can be exploited by an adversary under this category: the frequency of an encoding, the length of an encoding, and the similarity neighbourhood of an encoding.

- **Frequency Vulnerability and Assignability**:

  In frequency based vulnerability and assignability, an encoding e_i that is frequency vulnerable, can be assigned to one or more values v_a in the set of plaintext values $\mathbf{V}$ because their corresponding frequencies, $freq(e_i)$ and $freq(v_a)$, are the same or similar. Formally, for an encoding $e_i \in \mathbf{H}$ and the set $\mathbf{m}_f = \{v_a : v_a \in \mathbf{V}, |freq(e_i) - freq(v_a)| \leq \varepsilon\}$ of vulnerable plaintext values that each has a frequency difference with at most a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the pair (e_i, v_a) is (ε, k)-frequency assignable if $1 \leq |\mathbf{m}_f| < k$.

  Note that depending on the sizes of the plaintext and encoded databases, $|\mathbf{D}^s|$ and $|\mathbf{D}^e|$, the calculated frequencies of plaintext and of encoded values will

need to be normalised. Such normalisation will allow accurate assignment between plaintext and encoded values.

- **Length Vulnerability and Assignability**:

  In length based vulnerability and assignability, an encoding e_i is assigned to one or more values v_a in the plaintext values $\mathbf{V}$ because their corresponding lengths, $len(e_i)$ and $len(v_a)$, are the same or similar. Formally, for an encoding $e_i \in \mathbf{H}$ and the set $\mathbf{m}_l = \{v_a : v_a \in \mathbf{V}, |len(e_i) - len(v_a)| \leq \varepsilon\}$ of plaintext values that each has a length difference of at most a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the pair (e_i, v_a) is (ε, k)-length assignable if $1 \leq |\mathbf{m}_l| < k$.

  Depending on the PPRL encoding technique used, the length calculation as well as the actual lengths of the encodings can be different from the plaintext value length calculations, and their corresponding length values. For instance with BF encoding [177], the length of a BF can be considered as the Hamming weight (number of 1-bits) of the corresponding bit vector. The Hamming weight of a BF is different from the number of q-grams extracted from the plaintext value that is encoded in that BF, because each q-gram (token substring) is encoded using m hash functions resulting in up to m bits being set by each q-gram [177]. Therefore, when conducting length assignment, the relative lengths of plaintext values and encodings need to be adjusted. One possibility is to normalise the length values of both plaintext values and encodings, and then use such normalised length values to assign encodings and plaintext values.

- **Similarity Neighbourhood Vulnerability and Assignability**:

  In similarity neighbourhood based vulnerability and assignability, a vulnerable encoding e_x is assigned to one or more vulnerable values v_a in the set of plaintext values $\mathbf{V}$ because their corresponding neighbourhoods $\mathbf{e}_x$ and $\mathbf{v}_a$ in the corresponding similarity graphs are the same or similar. The two neighbourhoods $\mathbf{e}_x$ and $\mathbf{v}_a$ are compared using the sets $feat(\mathbf{e}_x)$ and $feat(\mathbf{v}_a)$ of features calculated using the corresponding neighbourhoods. Formally, for a unique encoding $e_i \in \mathbf{H}$ and the set $\mathbf{m}_n = \{v_a : v_a \in \mathbf{V}, dist(feat(\mathbf{e}_x), feat(\mathbf{v}_a)) \leq \varepsilon\}$ of vulnerable plaintext values that each has a distance between their neighbourhood features with at most a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the pair (e_x, v_a) is (ε, k)-neighbourhood assignable if $1 \leq |\mathbf{m}_n| < k$. Here $dist()$ is the function that measures the distance between the two feature vectors.

5.6.2 Vulnerability and Assignability of a Pair of Encodings

Two types of characteristics can be exploited by an adversary under this category: the co-occurrence of two (or more) encodings and the similarity between a pair of encodings.

- **Co-occurrence Vulnerability and Assignability**:

  In co-occurrence based vulnerability and assignability, a set of two or more vulnerable encodings (from the same record) $\mathbf{e}_x$ are assigned to one or more sets of vulnerable values $\mathbf{v}_a$ in the plaintext values $\mathbf{V}$ because their corresponding co-occurrence frequencies, $freq(\mathbf{e}_x)$ and $freq(\mathbf{v}_a)$, are the same or similar. Formally, for a set of unique encodings $\mathbf{e}_x \in \mathbf{H}$ and the set $\mathbf{M}_c = \{\mathbf{v}_a : \mathbf{v}_a \in \mathbf{V}, |freq(\mathbf{e}_x) - freq(\mathbf{v}_a)| \leq \varepsilon\}$ of sets of vulnerable plaintext values that each has a co-occurring frequency difference with at most a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the pair of sets $(\mathbf{e}_x, \mathbf{v}_a)$ is (ε, k)-co-occurrence assignable if $1 \leq |\mathbf{M}_c| < k$.

- **Similarity Vulnerability and Assignability**:

  In similarity based vulnerability and assignability, a pair of vulnerable encodings (e_i, e_j) is assigned to one or more pairs of vulnerable values (v_a, v_b) in the plaintext values $\mathbf{V}$ because their corresponding pairwise similarities, $sim(e_i, e_j)$ and $sim(v_a, v_b)$, are the same or similar. Formally, for a pair of unique encodings (e_i, e_j) where $e_i \in \mathbf{H}$ and $e_j \in \mathbf{H}$, and the set $\mathbf{m}_s = \{(v_a, v_b) : v_a \in \mathbf{V}, v_b \in \mathbf{V}, |sim(e_i) - sim(v_a)| \leq \varepsilon\}$ of vulnerable plaintext pairs of values that each has a similarity difference with at most a tolerance ε, with k and $\varepsilon \geq 0$ being two privacy parameters, then the pair of pairs $[(e_i, e_j), (v_a, v_b)]$ is (ε, k)-similarity assignable if $1 \leq |\mathbf{m}_s| < k$.

  Similar to length assignment, depending on the encoding technique used, the similarities calculated between encoded values can be different from the similarities calculated between plaintext values [214]. Therefore, when conducting the similarity assignment the plaintext similarities should be adjusted with respect to encoding similarities, or vice versa.

5.7 Relevance of Plaintext and Encoded Vulnerabilities

It is important to note that not all five of the above discussed vulnerabilities are equally applicable to QID values, tokens, and token substrings, or to all PPRL encoding techniques. In Table 5.3 we show the applicability of each vulnerability for different plaintext values and for five existing PPRL encoding techniques. For instance, length vulnerability cannot be assessed for token substrings as opposed to tokens, assuming token substrings (such as q-grams) have a fixed predefined length. Similarly, length vulnerability cannot be assessed for the MMK encoding technique [163] because this technique generates a set of fixed length hash codes for each record in the sensitive database $\mathbf{D}^s$.

In general, an adversary gains less information by reidentifying a token substring (such as a q-gram) compared to reidentifying a token (using the same vulnerability assessment), because a q-gram is only a part of a token. However, if a certain q-gram (token substring) is unique in the sensitive database (if the q-gram occurs only

Table 5.3: The application of vulnerability assessments for different plaintext values and encoding techniques used in the context of PPRL. In this table we show the applicability of each vulnerability for plaintext values $\mathbf{Q}^s$, **T**, and **S** (as illustrated in Table 5.2), and for five existing encoding techniques in PPRL, statistical linkage key (SLK) [110], BFs [177], TMH [186], MMK [163], and 2SH [158]. A ✓ indicates that the corresponding vulnerability is being exploited by an existing PPRL attack, whereas a – indicates that it does not. We also use (✓) to indicate that the corresponding encoding technique might have that vulnerability, but this has not yet been explored or discussed in the literature.

Vulnerability	**Plaintext**			**Encoding**				
	$\mathbf{Q}^s$	**T**	**S**	SLK	BF	TMH	MMK	2SH
Frequency	✓	✓	✓	✓	✓	✓	✓	✓
Length	✓	✓	–	–	✓	(✓)	–	(✓)
Co-occurrence	✓	✓	✓	–	✓	(✓)	✓	(✓)
Similarity	✓	✓	–	–	(✓)	(✓)	(✓)	(✓)
Neighbourhood	✓	✓	–	–	✓	✓	(✓)	✓

in one token in the sensitive database, for example), reidentifying that particular q-gram could lead to the reidentification of the token, and potentially the reidentification of the corresponding entity itself. Similarly, reidentifying a token will provide an adversary with less information compared to reidentifying a QID value. For instance, the QID value 'Hans Dieter' has two tokens and eight q-grams when $l_q = 2$. By reidentifying a single token an adversary is able to gain more information than she can gain from reidentifying a single q-gram, while she can gain even more from reidentifying an entire QID value. However, as we mentioned before most QID values will only contain a single token and therefore reidentifying a token in such a case will lead to the reidentification of the QID value itself.

In Table 5.4, we show the different vulnerabilities exploited by different attack methods in PPRL. As can be seen, most of these attacks have exploited the frequency vulnerability and assignability. However, no PPRL attack has been proposed so far that exploits the similarity vulnerability and assignability. This is because in real-world situations there are hardly any similarities between pairs of encodings or plaintext values that occur uniquely within the encoded or plaintext databases, especially for large databases. In Chapter 4 we discussed how each of these attack methods exploit the vulnerabilities in sensitive databases to reidentify encoded QID values. Note that the last three attacks mentioned in Table 5.4 corresponds to the attacks we propose in Chapters 6 to 8.

5.8 Analysis of Existing Attacks on PPRL

In this section we briefly discuss how existing privacy attacks on PPRL exploit the vulnerabilities discussed in this chapter to reidentify encoded QID values. We further

Table 5.4: Vulnerabilities and the corresponding assignabilities (as we discuss in Section 5.6) in encoded and plaintext databases exploited by various attacks on PPRL. As can be seen, only four of the five characteristics have been explored by these attacks.

Attack method	**Vulnerability and Assignability**				
	Frequency	Length	Co-occurrence	Similarity	Neighbourhood
Kuzu et al. (PET 2011) [121]	✓	✓	✓	–	–
Niedermeyer et al. (JPC 2014) [147]	✓	–	–	–	–
Kroll and Steinmetzer (BIOSTEC 2015) [119]	✓	–	✓	–	–
Christen et al. (PAKDD 2017) [42]	✓	–	–	–	–
Mitchell et al. (IJBDI 2017) [139]	–	–	✓	–	–
Culnane et al. (arXiv 2017) [46]	–	–	–	–	✓
Christen et al. (PAKDD 2018) [43]	✓	–	✓	–	–
Christen et al. (TKDE 2018) [41]	✓	✓	–	–	–
Vidanage et al. (ICDE 2019) [216]	✓	–	✓	–	–
Vidanage et al. (IJPDS 2020) [215]	✓	–	–	–	–
Vidanage et al. (CIKM 2020) [214]	–	–	–	–	✓

discuss the behaviours of the existing attacks when such vulnerabilities do not exist in encoded databases.

As we discussed in Section 5.4, an attack on an encoded database consists of two steps, attribute reidentification and identity reidentification (which lead to attribute and identity disclosures). Attribute reidentification occurs when an adversary exploits one or more vulnerabilities in the encoded database, as we discussed in Section 5.6. Most of the existing attacks exploit frequency and co-occurrence vulnerabilities of encodings [41, 42, 43, 119, 121, 147] whereas few attacks exploit length and similarity neighbourhood vulnerabilities [41, 46].

For instance, the attack proposed by Kuzu et al. [121] explore the frequencies of BFs, lengths of 1-bit patterns in BFs, and co-occurrences of bit positions in BFs in order to assign those BFs with plaintext values with similar characteristics. The attack proposed by Kroll and Steinmetzer [119] first aligns the frequent BFs with frequent QID values and then use an optimisation algorithm to find the maximum correlation between q-gram pairs and bit pattern pairs that can encode a single q-gram. The graph based attack we propose in Chapter 8, aligns encoded records with plaintext records based on the uniqueness of the similarity neighbourhoods of records. Once reidentified, using the uniqueness of the attributes the adversary can then disclose the identities of people who are represented by those attributes.

In Chapter 4 we provided detailed descriptions about the methodologies used by the existing attack methods on PPRL. When analysing those methodologies it becomes apparent that the basic building blocks of all existing attack methods include exploiting at least one of the vulnerabilities that occur in both encodings $e \in \mathbf{H}$ and plaintext values $v \in \mathbf{V}$. As we discussed above, $\mathbf{V}$ can either be QID values, $\mathbf{Q}^s$, tokens, $\mathbf{T}$, or token substrings, $\mathbf{S}$. Therefore, for an attack to be successful in reidentifying attribute values and real-world entities, both the encoded and plaintext values, $\mathbf{E}$ and $\mathbf{Q}^s$, need to be vulnerable.

Furthermore, if no encoding $e \in \mathbf{H}$ or no plaintext value $v \in \mathbf{V}$ is (ε, k)-vulnerable, and no encoding and plaintext value pair (e, v), is (ε, k)-assignable then we can define the encoded list of values $\mathbf{E}$ as secure with regard to the lists of plaintext values ($\mathbf{Q}^s$, $\mathbf{T}$, or $\mathbf{S}$), the corresponding encoding function, $encode()$, and its parameters, $\mathbf{p}$, used, and the parameters ε and k. This is because without the vulnerabilities that occur in both the encoded and plaintext values, $\mathbf{E}$ and $\mathbf{Q}^s$, no existing attack on PPRL would be successful in reidentifying encoded attribute values correctly.

For instance, the initial steps of the attacks proposed by Kuzu et al. [121], Niedermeyer et al. [147], Kroll and Steinmetzer [119], Christen et al. [41, 42, 43] include the alignment of frequent BFs to frequent QID values or frequent bit patterns to frequent q-grams. The remaining steps of the attacks depend on the accuracy of these alignments. If no frequent BFs, bit patterns, QID values, or q-grams can be found in their corresponding databases, then the attacks would fail. The attack proposed by Culnane et al. [46] matches encodings with plaintext QID values based on the uniqueness of their similarity neighbourhoods. If no such unique similarity neighbourhoods can be identified, then the attack would fail. The attack proposed by Mitchell et al [139], on the other hand, is a special type of an attack where it is assumed that the adversary has access to all encoding parameters. Even though it does not depend on the co-occurrence frequencies of QID values or q-grams, it depends on the co-occurrence of q-grams being connected to each other such that the end character of one q-gram will be the first character of another. If this type of co-occurrence does not exist in q-grams the attack would fail.

5.9 Experimental Evaluation

We conducted an experimental evaluation of our proposed vulnerability assessment framework for PPRL using a real-world database and a synthetic database. We measured the different types of vulnerabilities in the plaintext databases, and used five different PPRL encoding techniques to measure the vulnerabilities associated with encoded databases.

5.9.1 Databases and Evaluation Setup

First we used the North Carolina Voter Registration (NCVR) database[3], where we used one snapshot collected in December 2020 to randomly sample a subset of 100,000 records. These records contains personal details of North Carolina voters. The second database we used is a European census (EURO-CS) database[4] generated to represent real observations of the decennial census. The database contains personal details of 25,343 fictitious people. We describe the characteristics of the EURO-CS database in detail in Chapter 9. We used different combinations of the attributes first name, last name, street address, city, and zipcode, to generate dif-

[3] http://dl.ncsbe.gov/data/

[4] https://ec.europa.eu/eurostat/cros/content/job-training_en

Table 5.5: The vulnerability results of the plaintext QID values, $\mathbf{Q}^s$, tokens, $\mathbf{T}$, and token substrings, $\mathbf{S}$, in the NCVR and EURO-CS databases calculated for different k and ε parameter settings and for different attributes. Results are shown as the percentages of vulnerable values.

			FirstName				*LastName*				*StreetAddress*				*City/Zipcode*			
		$k=$	**10**		**20**		**10**		**20**		**10**		**20**		**10**		**20**	
		$\varepsilon=$	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%
NCVR	Freq	$\mathbf{Q}^s$	3.42	0.38	4.64	0.67	1.22	0.13	1.71	0.28	0.04	0.02	0.04	0.03	67.9	2.76	100	5.80
		$\mathbf{T}$	3.34	0.41	4.70	0.68	1.14	0.14	1.71	0.26	1.42	0.06	2.16	0.08	70.9	2.92	97.3	7.17
		$\mathbf{S}$	65.1	10.6	74.4	12.9	80.2	7.12	87.4	13.7	62.7	5.55	68.7	9.02	100	13.0	100	22.4
	Len	$\mathbf{Q}^s$	0.06	0.06	0.40	0.40	0.04	0.04	0.11	0.11	0.03	0.03	0.06	0.06	3.18	3.18	3.18	3.18
		$\mathbf{T}$	0.18	0.18	0.18	0.18	0.05	0.05	0.05	0.05	0.05	0.05	0.13	0.13	0.93	0.93	7.04	7.04
	Co-occur	$\mathbf{Q}^s$	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
		$\mathbf{T}$	4.85	4.85	4.85	4.85	1.09	1.09	1.09	1.09	0.11	0.01	0.18	0.03	95.2	10.0	100	12.9
		$\mathbf{S}$	5.97	0.19	8.68	0.54	2.13	0.06	3.57	0.10	2.11	0.02	3.42	0.02	18.8	0.40	28.9	0.77
	Sim	$\mathbf{Q}^s$	0.02	0	0.03	0.01	0.01	0	0.02	0	0	0	0.01	0	7.35	2.0	16.7	3.65
		$\mathbf{T}$	0.01	0	0.03	0.01	0.01	0	0.02	0	0.01	0	0.01	0	3.65	1.77	8.43	4.26
	Neigh	$\mathbf{Q}^s$	100	63.0	100	84.9	100	34.7	100	55.4	100	28.1	100	48.6	100	100	100	100
		$\mathbf{T}$	100	64.3	100	85.9	100	38.4	100	59.8	100	34.4	100	51.9	100	100	100	100
EURO -CS	Freq	$\mathbf{Q}^s$	13.6	4.15	15.7	7.9	11.9	8.16	13.6	10.40	0.50	0.50	1.75	1.75	19.0	3.0	69.1	11.2
		$\mathbf{T}$	13.6	4.15	15.7	7.93	11.9	8.16	13.6	10.4	100	6.95	100	8.56	19.0	2.98	69.1	11.2
		$\mathbf{S}$	72.2	19.0	93.0	31.2	76.9	15.5	94.7	26.1	100	12.9	100	23.5	98.3	4.01	100	15.0
	Len	$\mathbf{Q}^s$	0.37	0.37	0.37	0.37	0.87	0.87	0.87	0.87	0	0	0	0	1.32	1.32	1.32	1.32
		$\mathbf{T}$	0.37	0.37	0.37	0.37	0.87	0.87	0.87	0.87	21.93	21.93	27.8	27.8	1.32	1.32	1.32	1.32
	Co-occur	$\mathbf{Q}^s$	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
		$\mathbf{T}$	–	–	–	–	–	–	–	–	8.75	1.69	11.7	3.37	–	–	–	–
		$\mathbf{S}$	10.4	0.56	15.4	1.45	13.0	0.50	19.2	1.21	8.49	0.17	14.2	0.28	9.41	1.13	22.6	2.09
	Sim	$\mathbf{Q}^s$	0.09	0.05	0.38	0.21	0.43	0.27	1.14	0.98	0.01	0	0.02	0.01	1.72	1.72	4.72	4.72
		$\mathbf{T}$	0.09	0.05	0.38	0.21	0.43	0.27	1.14	0.98	6.57	6.57	6.57	6.57	1.72	1.72	4.72	4.72
	Neigh	$\mathbf{Q}^s$	100	97.9	100	99.6	98.6	93.0	100	95.0	99.5	52.5	100	70.0	72.6	45.7	82.1	58.9
		$\mathbf{T}$	100	97.9	100	99.6	98.6	92.7	100	95.0	100	100	100	100	72.6	45.7	82.1	58.9

ferent instances of encoded databases using the four encoding technique BF [177], TMH [186], MMK [163], and 2SH [158], as we described in Chapter 2. These selected attributes are commonly used in PPRL evaluation studies [209]. For SLK [110] encoding, however, we used the four attributes first name, last name, date of birth, and gender as SLK encoding is based on those attributes.

For both the plaintext and encoding databases, we analysed all five vulnerabilities for the three plaintext value lists, QIDs $\mathbf{Q}^s$, tokens $\mathbf{T}$, and token substrings $\mathbf{S}$, and for the five encoding techniques, respectively. We used $k = [10, 20]$ and $\varepsilon = [0\%, 1\%]$ for the privacy parameter settings, where we calculated the actual ε values from the given percentage values based on the minimum and maximum values of the corresponding vulnerability characteristic (such as the minimum and maximum frequency or length of values). For token substrings we used q-grams of length $l_q = 2$ for all the experiments. For each PPRL encoding technique, we used a single combination of parameter settings to encode plaintext values. These settings follow earlier work and/or seminal work of the corresponding PPRL encoding methods [110, 158, 163, 177, 186, 209].

Table 5.6: The co-occurrence vulnerability results of the plaintext QID values, $\mathbf{Q}^s$, tokens, **T**, and token substrings, **S** in the NCVR and EURO-CS databases calculated for different k and ε parameter settings and for different attribute combinations. Results are shown as the percentages of vulnerable values.

			FirstName, LastName				*LastName, City/Zipcode*				*FirstName, LastName StreetAddress*				*FirstName, LastName StreetAddress, City/Zipcode*			
		$k =$	**10**		**20**		**10**		**20**		**10**		**20**		**10**		**20**	
		$\varepsilon =$	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%
NCVR	Co-occur	$\mathbf{Q}^s$	0.02	0.02	0.05	0.05	0.10	0.05	0.15	0.07	0.01	0.01	0.02	0.02	0.03	0.01	0.04	0.01
		T	0.02	0.02	0.03	0.03	0.20	0.02	0.31	0.03	0.05	0.01	0.08	0.01	0.04	0	0.08	0.01
		S	2.74	0.05	4.35	0.11	3.49	0.05	5.10	0.10	2.27	0.01	3.86	0.02	3.01	0.02	4.92	0.03
EURO -CS	Co-occur	$\mathbf{Q}^s$	0.04	0.04	0.04	0.04	0.09	0.09	0.09	0.09	0.01	0.01	0.03	0.03	0.01	0.01	0.01	0.01
		T	0.04	0.04	0.04	0.04	0.09	0.09	0.09	0.09	0.63	0.04	0.83	0.09	0.33	0.03	0.39	0.05
		S	3.27	0.07	5.09	0.10	0.91	0.01	1.92	0.04	2.98	0.03	4.71	0.07	1.36	0.01	2.07	0.03

5.9.2 Results and Discussion

In Table 5.5 we show the vulnerability percentages of plaintext values calculated for both the databases NCVR and EURO-CS. As can be seen, with larger values of k and smaller values of ε the percentages of vulnerable values increase for frequency, co-occurrence, and neighbourhood vulnerabilities. This is because when k increases the threshold for the size of the value set that is assumed to be distinguishable by an adversary for a given vulnerability increases. On the other hand, with decreasing values for ε, the difference between plaintext values (in a given vulnerability) that are assumed to be distinguishable by an adversary decreases. In both of these instances, the number of vulnerable plaintext values in the database increases.

For the length, co-occurrence, and similarity vulnerabilities, however, the percentages of vulnerable values stayed approximately the same for all three lists $\mathbf{Q}^s$, **T**, and **S**. This is because there are no more vulnerable values in this particular plaintext database when considering the parameter values $k = 20$ and $\varepsilon = 0\%$ compared to when $k = 10$ and $\varepsilon = 1\%$. It is, however, worth noting that with even larger values of k, such as $k = [50, 100]$, there will likely be more vulnerable plaintext values. As shown in Table 5.5 the co-occurrence vulnerability cannot be calculated for the lists $\mathbf{Q}^s$ and **T** when only one attribute is considered, except for the list **T** with the attribute street address. This is because, unlike other attributes, QID values in street address can be divided into multiple tokens, as we discussed in Section 5.4.

In Table 5.6 we show the co-occurrence vulnerability for different plaintext attribute combinations. As can be seen, the sets of substrings in the list **S** generally resulted in higher vulnerability percentages compared to the other two sets of plaintext values, $\mathbf{Q}^s$ and **T**. This is because substrings (bigrams in these experiments) have more co-occurrence frequency information that can be uniquely identified with respect to the defined k and ε values compared to QIDs and tokens.

In Tables 5.7 to 5.9 we illustrate the vulnerability and assignability percentage results for the five encoding techniques we discussed in Chapter 2. Similar to the plaintext vulnerability results, it can be seen that with increasing values for k and

Table 5.7: The vulnerability results of the encoded values in **E** calculated for different k and ε parameter settings and for encoding techniques BF [177], TMH [186], and 2SH [158]. Results are shown as the percentages of vulnerable values and for both the NCVR and EURO-CS databases.

			FirstName				*FirstName, LastName*				*FirstName, LastName, StreetAddress*				*FirstName, LastName, StreetAddress, City/Zipcode*			
		$k=$	**10**		**20**		**10**		**20**		**10**		**20**		**10**		**20**	
		$\varepsilon=$	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%
NCVR	Freq	**BF**	3.43	0.38	4.67	0.67	0.02	0.02	0.05	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
		TMH	3.43	0.38	4.67	0.67	0.02	0.02	0.05	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
		2SH	3.43	0.38	4.67	0.67	0.02	0.02	0.05	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	Len	**BF**	1.78	0.30	3.63	0.62	0.24	0.04	0.63	0.05	0.25	0.03	0.60	0.06	0.25	0.03	0.58	0.06
		TMH	0.88	0.25	1.31	0.55	0.07	0.02	0.14	0.04	0.08	0.02	0.16	0.05	0.08	0.02	0.14	0.04
		2SH	1.62	0.30	2.99	0.53	0.19	0.03	0.58	0.04	0.31	0.03	0.66	0.07	0.30	0.02	0.56	0.04
	Co-occur	**BF**	–	–	–	–	0.02	0.02	0.05	0.05	0.01	0.01	0.02	0.02	0.02	0.01	0.04	0.01
	Sim	**BF**	1.65	0	3.26	0.01	0.09	0	0.18	0	0.13	0	0.26	0	0.08	0	0.15	0
		TMH	0.01	0	0.02	0	0	0	0	0	0	0	0.01	0	0.01	0	0.01	0
		2SH	1.40	0	2.62	0	0.11	0	0.22	0	0.45	0	0.91	0	0.10	0	0.17	0
	Neigh	**BF**	100	49.3	100	70.6	100	3.15	100	6.23	100	3.57	100	6.24	99.9	4.3	99.9	8.4
		TMH	69.3	17.5	87.5	25.5	100	10.3	100	18.9	100	8.29	100	16.6	99.9	47.1	99.9	59.3
		2SH	100	47.2	100	77.4	100	9.5	100	18.3	100	12.7	100	24.9	100	74.5	100	80.2
EURO -CS	Freq	**BF**	13.5	5.96	15.7	10.0	0.04	0.04	0.04	0.04	0	0	0	0	0	0	0	0
		TMH	13.5	5.96	15.7	10.0	0.04	0.04	0.04	0.04	0	0	0	0	0	0	0	0
		2SH	13.5	5.96	15.7	10.0	0.04	0.04	0.04	0.04	0	0	0	0	0	0	0	0
	Len	**BF**	9.79	1.62	21.4	6.70	0.76	0.13	1.88	0.30	0.69	0.10	1.88	0.19	0.57	0.14	1.73	0.21
		TMH	5.77	1.34	11.3	4.30	0.42	0.13	1.04	0.26	0.34	0.09	0.77	0.29	0.32	0.15	0.81	0.24
		2SH	3.42	0.79	11.6	1.76	0.92	0.08	1.91	0.32	0.83	0.11	1.65	0.21	0.78	0.10	1.59	0.23
	Co-occur	**BF**	–	–	–	–	0.04	0.04	0.04	0.04	0.01	0.01	0.03	0.03	0.01	0.01	0.01	0.01
	Sim	**BF**	32.3	0.04	47.3	0.16	0.44	0	0.94	0	0.77	0	1.86	0	3.05	0	6.43	0
		TMH	2.30	0.10	8.36	0.17	0	0	0.01	0	0	0	0	0	0	0	0.01	0
		2SH	21.8	0.11	36.6	0.18	0.51	0	1.14	0	1.47	0	3.55	0	5.03	0	9.50	0
	Neigh	**BF**	100	98.4	100	100	100	30.5	100	52.0	100	15.2	100	25.7	100	34.7	100	50.8
		TMH	100	98.2	100	100	100	20.7	100	33.5	100	18.3	100	28.3	100	34.6	100	52.9
		2SH	100	98.5	100	100	100	12.	100	22.7	100	19.6	100	28.2	100	45.8	100	69.6

decreasing values for ε, the percentages of vulnerable encodings increase. As we showed in Table 5.3, not all vulnerability aspects can be assessed for each encoding technique. For instance, co-occurrence vulnerability for BF encoding can only be calculated if attribute-level BF encoding [177] is used with attribute combinations that consist of two or more attributes. Similarly, for SLK encoding only the frequency vulnerability, while for MMK encoding only the frequency and co-occurrence vulnerabilities can be assessed.

As can be seen from Table 5.7, the similarity neighbourhood vulnerabilities of encodings are higher compared to the other vulnerabilities. This is because the encodings are more distinguishable when using their similarity neighbourhood feature values compared to their frequencies, lengths, similarities, or co-occurrences. When three or four attributes (especially including street address) are encoded, no or only a very small number of encodings were found to be vulnerable because each encoded value was unique in those instances. In Table 5.8 we show the assignability

Table 5.8: The assignability results of the encoded values in **E** calculated for different k and ε parameter settings and for encoding techniques BF [177], TMH [186], and 2SH [158]. Results are shown as the percentages of vulnerable values and for both the NCVR and EURO-CS databases.

			FirstName				*FirstName, LastName*				*FirstName, LastName, StreetAddress*				*FirstName, LastName, StreetAddress, City/Zipcode*			
		$k=$	**10**		**20**		**10**		**20**		**10**		**20**		**10**		**20**	
		$\varepsilon=$	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%	0%	1%
NCVR	Freq	**BF**	3.43	0.38	4.67	0.67	0.02	0.02	0.05	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
		TMH	3.43	0.38	4.67	0.67	0.01	0.01	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
		2SH	3.43	0.38	4.67	0.67	0.02	0.02	0.05	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	Len	**BF**	0.01	0.02	0.04	0.12	0	0.01	0	0.01	0	0.02	0	0.03	0	0.01	0	0.02
		TMH	0.01	0.04	0.01	0.04	0	0	0	0	0	0	0	0	0	0	0	0
		2SH	0.01	0.03	0.02	0.08	0.01	0.02	0.01	0.02	0	0.02	0	0.04	0	0.01	0	0.02
	Co-occur	**BF**	–	–	–	–	0.01	0.01	0.02	0.02	0.0	0.0	0.01	0.01	0.0	0.0	0.0	0.0
	Sim	**BF**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		TMH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		2SH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Neigh	**BF**	0	0.01	0	0.02	0	0	0	0	0	0.01	0	0.01	0	0	0	0
		TMH	0	0.12	0	0.12	0	0	0	0	0	0	0	0.01	0	0	0	0
		2SH	0	0.02	0	0.02	0	0	0	0.02	0	0	0	0	0	0	0	0
EURO -CS	Freq	**BF**	13.5	5.96	15.70	10.02	0.04	0.04	0.04	0.04	0	0	0	0	0	0	0	0
		TMH	13.5	5.96	15.70	10.02	0.04	0.04	0.04	0.04	0	0	0	0	0	0	0	0
		2SH	13.5	5.96	15.70	10.02	0.04	0.04	0.04	0.04	0	0	0	0	0	0	0	0
	Len	**BF**	0.05	0.14	0.05	0.14	0.02	0.02	0.02	0.11	0.01	0.02	0.01	0.08	0.01	0.02	0.01	0.02
		TMH	0.05	0.09	0.05	0.18	0.02	0.02	0.02	0.03	0.01	0.03	0.01	0.03	0.01	0.01	0.01	0.01
		2SH	0.05	0.05	0.05	0.05	0.01	0.02	0.01	0.05	0.01	0.05	0.01	0.05	0.01	0.04	0.01	0.04
	Co-occur	**BF**	–	–	–	–	0.03	0.03	0.03	0.03	0.0	0.0	0.03	0.03	0.01	0.01	0.01	0.01
	Sim	**BF**	0.01	0	0.05	0.09	0	0	0	0	0	0	0	0	0	0	0	0
		TMH	0	0	0	0.08	0	0	0	0	0	0	0	0	0	0	0	0
		2SH	0.01	0	0.12	0.10	0	0	0	0	0	0	0	0	0.01	0	0.02	0
	Neigh	**BF**	0	0.05	0	0.05	0	0.03	0	0.03	0	0.02	0	0.04	0	0	0	0.02
		TMH	0	0.19	0	0.19	0	0.01	0	0.01	0	0	0	0	0	0.01	0	0.01
		2SH	0	0.05	0	0.05	0	0.01	0	0.01	0	0.01	0	0.01	0	0.01	0	0.01

percentage results of encodings. As can be seen, the frequency assignabilities are high compared to the length, co-occurrence, similarity, and similarity neighbourhood assignabilities. With co-occurrence, the vulnerability of encodings are already low and thus it is expected to have low assignability percentages. The lengths of the encodings are also relatively different when compared with the lengths of the corresponding plaintext values. As shown in the previous studies [158] the similarity of two encodings will not be the same as the similarity of the two corresponding plaintext values in those encodings. Because of these reasons, the assignability results for length, similarity, and and similarity neighbourhood vulnerabilities will be low.

Furthermore, as shown in Table 5.9, the SLK encoding has no frequency vulnerable encodings with the EURO-CS database and a very few vulnerable encodings with the NCVR database, because almost all SLK encodings were unique (had a frequency of 1) for thes specific databases. While some match-keys in the MMK encoding had a small number ($< 1\%$) of frequency vulnerable encodings, no co-occurrence vulner-

Table 5.9: The vulnerability / assignability results of the encoded values in **E** calculated for different k and ε parameter settings and for encoding techniques SLK [110] and MMK [163]. Results are shown as the percentages of vulnerable values and for both the NCVR and EURO-CS databases.

				SLK	**MMK**			
					FirstName, StreetAddress	*StreetAddress, City/Zipcode*	*LastName, StreetAddress, City/Zipcode*	*FirstName, LastName, StreetAddress*
		k	ε					
NCVR	Freq	10	0%	0 / 0	0 / 0	0.04 / 0.01	0 / 0	0.01 / 0.01
			1%	0 / 0	0 / 0	0.02 / 0	0 / 0	0.01 / 0.01
		20	0%	0.2 / 0.2	0.02 / 0.02	0.04 / 0.01	0 / 0	0.01 / 0.01
			1%	0.2 / 0.2	0.02 / 0.02	0.03 / 0	0 / 0	0.01 / 0.01
EURO -CS	Freq	10	0%	0 / 0	0 / 0	0 / 0	0.05 / 0.05	0 / 0
			1%	0 / 0	0 / 0	0 / 0	0.05 / 0.05	0 / 0
		20	0%	0 / 0	0.04 / 0.04	0 / 0	0.05 / 0.05	0 / 0
			1%	0 / 0	0.04 / 0.04	0 / 0	0.05 / 0.05	0 / 0

able encodings were found for MMK. These vulnerability assessment results confirm the findings and observations in previous studies on PPRL attacks [41, 42, 214, 216].

It is worth noting that we expect both the vulnerability and assignability results for frequency, co-occurrence, and similarity neighbourhood to be higher with larger databases. This is because with a higher number of records, there will be more unique frequencies (both individual and co-occurring) and similarity neighbourhoods that can be exploited by an attack.

5.9.3 Recommendations

Based on our findings and observations we now provide a few recommendations related to assessing the vulnerabilities of sensitive databases for DOs who are working in the context of PPRL.

We first recommend DOs to conduct the plaintext vulnerability assessment using our proposed framework on their sensitive databases before participating in a PPRL project. We recommend to also assess the vulnerabilities associated with different PPRL encoding techniques using each database that is expected to be used in a PPRL project. This is because different databases will have different vulnerabilities due to their varying characteristics, such as the frequency or length distributions of their QID values. We also recommend to analyse different combinations of attributes with different PPRL techniques, because attribute combinations will also impact the vulnerabilities in an encoded database that might be exploited in an attack.

We recommend the owners of sensitive databases who plan to participate in a PPRL project to carefully assess the policies and regulations they are required to follow (as well as the privacy requirements of the linkage project) to identify suitable values for k and ε. Larger values of k and smaller values of ε will identify more plaintext values and encodings as being vulnerable. If no vulnerable plaintext or

encoded values are identified for large k and small ε then no values will be vulnerable for smaller k and large ε.

5.10 Summary

In this chapter we have presented a novel framework for assessing the vulnerabilities associated with both plaintext and encoded databases as used in the context of PPRL. Our proposed framework utilises five characteristics, namely frequency, length, co-occurrence, similarity, and similarity neighbourhood, of the databases, that can be exploited by an attack, to evaluate the vulnerabilities of these databases. We defined the terms (ε, k)-vulnerability and (ε, k)-assignability for plaintext and encoded values. We discussed how these definitions can be used to quantify the vulnerabilities in plaintext and encoded databases using the five characteristics we mentioned above. In an experimental study we applied our framework on a real-world and a synthetic database to illustrate that it can provide valuable information about the vulnerabilities of a plaintext database as well as a database encoded using different PPRL techniques. Given the increasing demand for secure PPRL encoding techniques and the ability to assess the vulnerabilities of the databases encoded using these techniques, such a systematic framework can be highly beneficial.

Chapter 6

A Frequency based Privacy Attack on Multiple Match-Key Encoding

In this chapter we propose our first frequency based privacy attack on a recently proposed privacy-preserving record linkage technique named multiple match-key (MMK) encoding. The aim of our attack is to identify weaknesses of the MMK encoding approach in order to evaluate its privacy guarantees.

In Section 6.1 we first describe how the quasi-identifying attribute values in a sensitive database are encoded and linked using the MMK encoding process, and we introduce our attack method. In Section 6.2 we next propose our attack on MMK encoding and describe the main five steps of the attack in detail. In Section 6.3 we analyse the computational complexity of our attack method, and in Section 6.4 we then experimentally evaluate our attack method using real-world databases and present two recommendations to prevent such a privacy attack. In Section 6.5 we discuss different practical aspects that need to be considered when using MMK encoding in real-world applications and finally in Section 6.6 we summarise our findings.

6.1 Introduction

The multiple match-key (MMK) encoding technique [163] is a recently proposed encoding technique for privacy-preserving record linkage (PPRL) which aims to provide privacy protection against frequency attacks, while achieving high linkage quality. As we describe next, the main idea of this technique is to generate distinct hash codes (called *match-key values*) using quasi-identifying (QID) attribute value combinations (called *match-keys*) and compare these match-key values across databases to identify matching pairs of records.

6.1.1 Multiple Match-Key Encoding

We now briefly describe the main steps of MMK encoding as proposed by Randall et al. [163]. Specific notation used throughout the chapter are shown in Table 6.1. We assume a sensitive database, $\mathbf{D}^s$, with n records, where each record contains k QID attributes. A match-key is a combination of QID attributes from $\mathbf{D}^s$. Given there

Table 6.1: Specific notation used throughout this chapter. For common notation used throughout the thesis, refer to the list of notation at the beginning of the thesis.

$\mathbf{mk}^e$, $\mathbf{mk}^p$	Lists of encoded and plaintext match-keys
mk^e, mk^p	An encoded and a plaintext match-key where $mk^e \in \mathbf{mk}^e$ and $mk^p \in \mathbf{mk}^p$
$\mathbf{H}$, $\mathbf{H}_{*j}$	Encoded matrix of hash codes, and a column in $\mathbf{H}$ which represents a match-key $mk^e \in \mathbf{mk}^e$
$\mathbf{M}$, $\mathbf{M}_{*j}$	Plaintext match-key matrix, and a column in $\mathbf{M}$ which represents a match-key $mk^p \in \mathbf{mk}^p$
$\mathbf{C}$, $\mathbf{C}_{i*}$, $\mathbf{C}_{*j}$	Correlation matrix, a row in $\mathbf{C}$ ($mk^p \in \mathbf{mk}^p$), and a column in $\mathbf{C}$ (correlation matric)
$\mathbf{f}^e$, $\mathbf{f}^p$	Frequency distributions of encoded and plaintext match-keys
k	Number of quasi-identifying (QID) attribute values in a database
i, j	ith row and jth column in a matrix
ε	Frequency range parameter
α_d, α	Correlation difference ratio and difference ratio limit between match-key frequency distributions
a_c	Overall average correlation
u_e, u_p	Number of unique values in an encoded and a plaintext match-key
d_c	Difference ratio between u_e and u_p
s_c	Weighted average of the two values a_c and d_c with weight w for a_c
Δ, Δ_t	Frequency difference and frequency difference threshold between a encoded match-key value and a plaintext match-key value

are k QID attributes in $\mathbf{D}^s$, there can be $2^k - 1$ combinations of at least one QID. A list of match-keys $\mathbf{mk}^e$ is first selected and then match-key values are created for each record $r \in \mathbf{D}^s$. These match-key values are encodings generated by hashing the concatenated values of each match-key $mk^e \in \mathbf{mk}^e$, as illustrated in Figure 6.1. An encoded database (consisting of hash codes) generated from $\mathbf{D}^s$ is denoted as $\mathbf{H}$, a matrix of hash codes. Essentially the matrix $\mathbf{H}$ is generated with n rows and m columns where $n = |\mathbf{D}^s|$ and $m = |\mathbf{mk}^e|$. A row in $\mathbf{H}$ corresponds to a record $r \in \mathbf{D}^s$ and each row contains a list of m hash codes. A column in $\mathbf{H}$ corresponds to a certain match-key $mk^e \in \mathbf{mk}^e$, where a given column contains the hash codes generated for all $r \in \mathbf{D}^s$. Hence a cell $\mathbf{H}_{ij}$ represents a hash code, h_{ij}, generated for the ith record using the jth match-key, where $1 \leq i \leq n$ and $1 \leq j \leq m$.

The list $\mathbf{mk}^e$ is determined using the probabilistic record linkage method proposed by Fellegi and Sunter [73], which employs the error characteristics of the linked databases. As described by Randall et al. [163], for each potential match-key a weight score w is calculated using the method proposed by Fellegi and Sunter [73]. For the MMK encoding, these weight scores w are then compared against a user-defined weight threshold, w_t. Only those match-keys that have a weight score of at least w_t are selected for $\mathbf{mk}^e$ to generate match-key values. The authors of the MMK encoding also recommended selecting match-keys with at least two QID attributes [163].

Sensitive plaintext database $\mathbf{D}^s$

RecordID	*FirstName* (F)	*LastName* (L)	*Gender* (G)	*YearOfBirth* (B)
$r1$	John	Smith	M	1985
$r2$	Peter	Miller		1991

Encoded database $\mathbf{D}^e$

RecordID	C_e^1 = (F + L + G)	C_e^2 = (F + L + B)	C_e^3 = (L + G + B)
$r1$	HMAC(JohnSmithM)	HMAC(JohnSmith1985)	HMAC(SmithM1985)
$r2$		HMAC(PeterMiller1991)	

Figure 6.1: An example encoding process assuming three QID value combinations (match-keys) where $|\mathbf{mk}_e| = 3$. The top table refers to the plaintext database QIDs before encoding, and the bottom table refers to the generated match-key values using HMAC [19]. Note that no match-key value (hash code) is generated if a match-key has a missing QID value.

Due to the potentially large number of match-keys that are possible for k QIDs ($2^k - (k+1)$ in total), Randall et al. [163] have employed a superset pruning approach to reduce the number of match-keys that are being selected in order to reduce the computational costs. For instance, if the match-key (*FirstName*, *LastName*) has a total score of at least w_t, then all the supersets of this match-key are pruned. This is because having a superset such as (*FirstName*, *LastName*, *Gender*) will not result in any additional matches compared to the matches already identified by the match-key (*FirstName*, *LastName*).

Once a list of match-keys $\mathbf{mk}^e$ is selected, match-key values are generated for each match-key using the corresponding QID attribute values of each record. To generate match-key values, a keyed cryptographic hash function, such as the hashed message authentication code (HMAC) [19] that uses a hash function and a secret key (known only to the owners of the databases that are being encoded for linkage) can be used. If one of the QID values used in a match-key is missing, then that match-key value is left empty. Given the list of selected match-keys $\mathbf{mk}^e$, each record in the database $\mathbf{D}^s$ will have up to $|\mathbf{mk}^e|$ match-key values assigned to it (less if the record has missing values in one or more of the QIDs used in $\mathbf{mk}^e$). We provide an example of this generation of match-key values in Figure 6.1. The match-key values for each $r \in \mathbf{D}^s$ are stored as lists (i.e. with an order), and only match-key values at the same position in these lists are compared between records. This will reduce the computational complexity of the comparison since one match-key value of a record will only be compared with one match-key value of another record.

In a linkage protocol, the owners of the sensitive databases to be linked first generate such a list of match-key values for each record in their database. Assuming two databases, $\mathbf{D}_A^s$ and $\mathbf{D}_B^s$, these databases are encoded into two matrices of match-key values (hash codes), $\mathbf{H}_A$ and $\mathbf{H}_B$, respectively. In the comparison and classification step, assumed to be conducted by a third party known as the linkage unit [210], the

corresponding match-key values (hash codes in the same column in $\mathbf{H}_A$ and $\mathbf{H}_B$) are then compared to classify each candidate record pair either as a match or non-match. Following Randall et al. [163], a record pair is classified as a match (assumed to refer to the same individual) if one or more of its match-key values are the same. If none of the corresponding match-key values of a record pair are the same, then the record pair will be classified as a non-match (assumed to refer to different individuals).

6.1.2 Contribution

In this chapter, we analyse the privacy guarantees of this MMK encoding technique using a privacy attack that aims to reidentify the plaintext values that were encoded into hash codes, using a known (publicly available) plaintext database. The attack tries to identify the QIDs used for match-keys by analysing the frequency distributions of individual match-key values. Using two large real-world databases, we show that the attack can reidentify plaintext values with an acceptable level of accuracy. We then propose two improvements as countermeasures for MMK encoding and discuss how these methods can strengthen the privacy guarantees of the MMK encoding technique and prevent this type of privacy attacks.

6.2 Frequency based Privacy Attack on MMK Encoding

We define the following terms which we use to describe the MMK encoding technique and the proposed privacy attack (as also summarised in Table 6.1). Assume a sensitive database $\mathbf{D}^s$ is encoded using the MMK encoding technique, which results in a matrix $\mathbf{H}$ of hash codes, as shown in Figure 1. Each column in $\mathbf{H}$ contains hash codes generated using a unique match-key, $mk^e \in \mathbf{mk}^e$. We refer to these columns as *encoded match-keys*.

As with existing attacks on PPRL [41, 43, 119], we assume the adversary has access to a plaintext database $\mathbf{D}^p$ with a frequency distribution of QID values similar to the encoded database $\mathbf{D}^s$. Such plaintext databases can be sourced externally (e.g. a telephone directory or voter database) or internally (if the attack is conducted by an insider who has access to some type of plaintext database that is similar to the encoded database). We then generate a list of match-keys, $\mathbf{mk}^p$, from the plaintext database $\mathbf{D}^p$ to compare against the encoded match-keys. We refer to these match-keys as *plaintext match-keys*. The resulting plaintext match-key matrix is denoted as M where each column is a plaintext match-key $mk^p \in \mathbf{mk}^p$. We assume that the plaintext database $\mathbf{D}^p$ contains some or all QIDs that are used for the generation of match-keys in $\mathbf{D}^s$. We also assume different levels of knowledge of the adversary, as we discuss in Section 6.4.2. The proposed privacy attack consists of five main steps:

1. Calculate the frequency distributions of encoded match-keys ($\mathbf{f}^e_j$) and plaintext match-keys ($\mathbf{f}^p_j$).

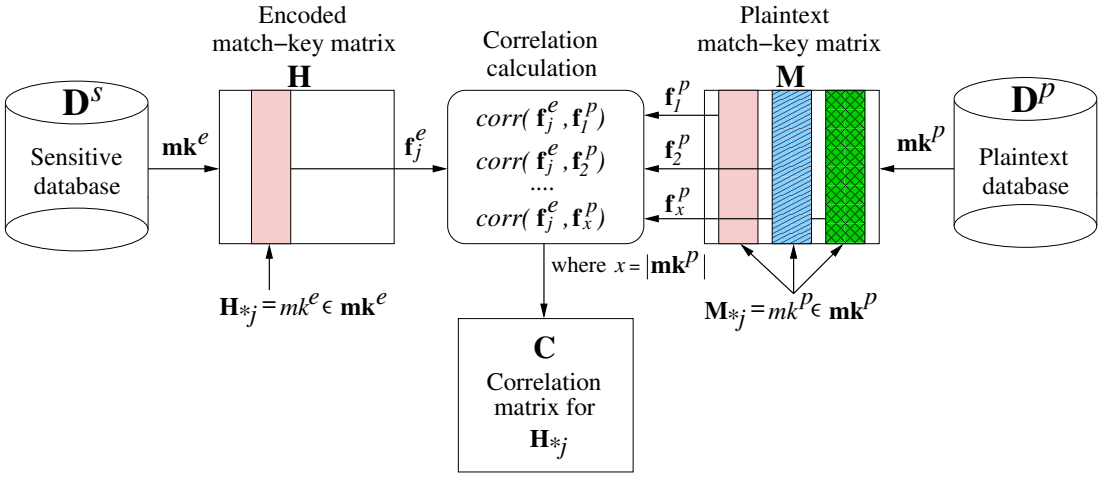


Figure 6.2: An overview of the first two steps of the proposed privacy attack. We first generate the plaintext match-key matrix, $\mathbf{M}$, and then we calculate the frequency distributions, $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$, of all encoded match-keys $\mathbf{H}_{*j} \in \mathbf{H}$ and plaintext match-keys $\mathbf{M}_{*j} \in \mathbf{M}$. Next, every possible pair of $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$ is compared using several correlation metrics. This results in a correlation matrix $\mathbf{C}$ for each encoded match-key $\mathbf{H}_{*j}$.

2. Calculate the correlations between the frequency distributions of each encoded match-key $\mathbf{H}_{*j} \in \mathbf{H}$ with all plaintext match-keys $\mathbf{M}_{*j} \in \mathbf{M}$, and for each $\mathbf{H}_{*j}$ build a correlation matrix, $\mathbf{C}$.

3. Select candidate plaintext match-keys for each encoded match-key $\mathbf{H}_{*j}$ by comparing correlation values in $\mathbf{C}$.

4. Assign highly correlated plaintext match-keys mk^p to each corresponding encoded match-key mk^e.

5. Reidentify plaintext values using frequency alignment of plaintext match-keys.

The first two steps of the attack are illustrated in Figure 6.2. We discuss these two steps in detail in the following.

6.2.1 Calculating Frequency Distributions

In the first step of the attack we calculate the frequency distributions of match-key values, $\mathbf{f}_j^e$, for each encoded match-key (i.e. column $\mathbf{H}_{*j} \in \mathbf{H}$) where $1 \leq j \leq |\mathbf{mk}^e|$. Since the match-key values of records are stored as lists (i.e. with an order), we are able to calculate frequency distributions of match-key values for each encoded match-key. Next, we generate different match-keys of the plaintext database $\mathbf{D}^p$. Assuming there are k QIDs in $\mathbf{D}^p$ (including some or all QIDs used for the encoded match-keys) we generate all match-keys from length two to k, assuming match-keys have only been generated from combinations of two or more QIDs [163]. For the resulting list of plaintext match-keys, $\mathbf{mk}^p$, for each record $r \in \mathbf{D}^p$, we generate a list of

plaintext match-key values using all $mk^p \in \mathbf{mk}^p$. This results in a matrix of plaintext match-key values, $\mathbf{M}$, where a column $\mathbf{M}_{*j} \in \mathbf{M}$ represents a plaintext match-key $mk^p \in \mathbf{mk}^p$. Then, for each column $\mathbf{M}_{*j}$, we calculate the frequency distribution, $\mathbf{f}_j^p$, of concatenated QID values in $\mathbf{mk}^p$ from $\mathbf{M}$.

For instance, if there are three QIDs *FirstName*, *LastName*, and *YearOfBirth* in the plaintext match-key database $\mathbf{M}$, we can generate four match-keys $\mathbf{M}_{*j}$ = (*FirstName* + *LastName*), (*FirstName* + *YearOfBirth*), (*LastName* + *YearOfBirth*), (*FirstName* + *LastName* + *YearOfBirth*). Here + is used to represent the string concatenation. We then calculate frequency distributions for all four match-keys. At the end of this first step the adversary has the frequency distributions of all encoded match-keys from $\mathbf{H}$ ($\mathbf{f}_j^e$) and all plaintext match-keys from $\mathbf{M}$ ($\mathbf{f}_j^p$).

6.2.2 Measuring the Correlation of Frequency Distributions

In the second step, we use a series of statistical tests to compare the frequency distributions of all plaintext match-keys, $\mathbf{f}_j^p$, with one given encoded match-key, $\mathbf{f}_j^e$. For each encoded match-key, $\mathbf{H}_{*j}$, we build a correlation matrix, $\mathbf{C}$ which contains all correlations calculated between each plaintext match-key, $\mathbf{M}_{*j}$, and that encoded match-key, $\mathbf{H}_{*j}$ (each pair of $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$), using different correlation metrics. Then by comparing these correlation values, we obtain the most similar plaintext match-keys for the current encoded match-key, $\mathbf{H}_{*j}$. In the following we discuss this process in more detail.

The statistical tests we use can be divided into two categories. The first consists of basic statistics which can be calculated for a distribution, while the second consists of correlation tests which can be used to compare two distributions. As for the basic statistics we calculate the mean, standard deviation, variance, and skewness [82] of two frequency distributions and calculate the absolute difference of those values for each pair of $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$. Then we employ the following correlation metrics to compare the two distributions $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$:

- Earth mover's distance (EMD) [130]: The EMD is used to measure the dissimilarity between two probability distributions over some feature space. Given one distribution as the baseline, EMD calculates the least amount of work needed to reach the second distribution.
- Kolmogorov-Smirnov (KS) test [227]: The KS test is used to measure the equality of two continuous probability distributions to inspect if one distribution is a sample of another distribution
- Pearson's correlation coefficient [48]: Given two continuous variables x and y, Pearson's correlation coefficient measures the correlation between these two variables. The relationship calculation is based on the method of co-variance.
- Spearman's rank correlation [48]: Spearman's Rank correlation measures the ranked correlation between two variables. Specifically, it calculates the strength and direction of association between two ranked variables.

- Relative entropy (Kullback-Leibler divergence) [93]: Relative entropy is used to measure how one probability distribution differs from another reference probability distribution. We use relative entropy to measure how much one frequency distribution is different from the second distribution.

- Histogram intersection [17]: Histogram intersection measures the similarity of two probability distributions by calculating the intersection of their histograms. For this calculation the two histograms of the probability distributions should be of the same length, or both distributions can be placed into a same number of bins using an appropriate binning technique such as equal depth binning. Given two histograms, $\mathbf{x}$ and $\mathbf{y}$, with b bins in each, the intersection of the two histograms can be calculated as:

$$I(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{b} min(x_i, y_i), \tag{6.1}$$

  where $min()$ outputs the minimum of x_i and y_i. To normalise the intersection value, we can divide $I(\mathbf{x}, \mathbf{y})$ by the average of the two histograms:

$$I(\mathbf{x}, \mathbf{y})_{norm} = \frac{2I(\mathbf{x}, \mathbf{y})}{\sum_{i=1}^{b} x_i + y_i}. \tag{6.2}$$

  In our attack, we use histogram intersection to calculate the intersection of the frequencies of distribution pairs $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$. The higher the value of $I(\mathbf{x}, \mathbf{y})_{norm}$, the more similar the two distributions are.

These six metrics are commonly used to compare the correlations between probability or frequency distributions. Given we are interested in measuring the correlations between two frequency distributions, we aim to identify correlated match-keys by employing these metrics. Furthermore, our evaluation using these metrics will help us to determine the usability of these correlation metrics in our attack method.

The different correlation values calculated using the above statistical tests will be added to a single vector, $\mathbf{c}$, for each pair of frequency distributions $\mathbf{f}_j^e$ and $\mathbf{f}_j^p$. A vector $\mathbf{c}$ will contain ten values, four values from the basic statistical metric differences and six values from the above discussed correlation metrics. For each encoded match-key, $\mathbf{H}_{*j}$, and its corresponding frequency distribution, $\mathbf{f}_j^e$, we obtain a set of correlation vectors by comparing $\mathbf{f}_j^e$ with the frequency distributions $\mathbf{f}_j^p$ of all possible plaintext match-keys in $\mathbf{M}$. This results in a correlation matrix $\mathbf{C}_j$ (for that particular encoded match-key $\mathbf{H}_{*j}$) where $\mathbf{C}_{i*}$ (ith row in the matrix) represents a correlation vector $\mathbf{c}$ calculated for a certain plaintext match-key and $\mathbf{C}_{*j}$ (jth column in the matrix) represents a vector of correlations for a certain correlation metric (i.e. one of the above ten tests). In practice it is important to understand which correlation metrics are robust when calculating correlations with different distributions. This aspect is outside the scope of this chapter and we considered the performance of all correlation metrics to be equal in our context. In the following steps we describe how we can

use the obtained correlation values to filter plaintext match-keys that are unlikely to match to encoded match-keys.

Frequency based filtering: Given there are k QIDs in $\mathbf{D}^p$, potentially there are $2^k - (k+1)$ QID combinations that we need to analyse (assuming each combination consists of at least two QIDs). Hence, we propose a filtering step to reduce this number of candidate plaintext match-keys by comparing their frequencies. We identify not possible plaintext match-keys through their top most frequent plaintext values. For instance, an encoded match-key with the highest frequency of 50 will not be compared with a plaintext match-key that has a highest frequency of 150. This is assuming that the two databases have a similar number of records and frequency distributions in their QID values. We define a frequency range parameter, ε, for the range of frequency values that can be used to filter plaintext match-keys. An encoded match-key with highest frequency of $max(\mathbf{f}^e_j)$, will only be compared with a plaintext match-key with highest frequency of $max(\mathbf{f}^p_j)$ if $max(\mathbf{f}^e_j) \leq max(\mathbf{f}^p_j) \leq max(\mathbf{f}^e_j) + \varepsilon$. Here $max()$ represents the top frequency (highest count) of that particular frequency distribution. The value for ε needs to be adjusted based on the encoded and plaintext database sizes, $|\mathbf{D}^s|$ and $|\mathbf{D}^p|$.

6.2.3 Filtering Candidate Plaintext Match-keys

In the third step of the attack we first normalise the correlation values in the matrix $\mathbf{C}$ for each statistical test using min-max normalisation. Here correlation values are normalised in such a way that the highest correlation will become 1.0 and the lowest correlation will become 0.0. However, the distance values (such as Earth mover's distance and relative entropy) are normalised in such a way that the highest value will become 0.0 and the lowest value will become 1.0. This ensures that all values obtained using the different statistical tests are in the same range and comparable.

Next, for each correlation metric, $\mathbf{C}_{*j}$, we sort the plaintext match-keys $\mathbf{mk}^p$ in descending order based on their normalised correlation values. We then select the top plaintext match-keys based on this sorted list. For each pair of plaintext match-keys, mk_x and mk_{x+1}, we calculate the difference ratio, α_d, of their normalised correlation values. We do this to filter the candidate plaintext match-keys that have the highest correlation values with small differences between them. Only plaintext match-keys that have a difference ratio below a user defined difference ratio limit of α will be considered. Assuming the correlation values of mk_x and mk_{x+1} are s_x and s_{x+1} respectively, the difference ratio between those two values, α_d, is calculated as:

$$\alpha_d = 2\left(\frac{s_x - s_{x+1}}{s_x + s_{x+1}}\right). \tag{6.3}$$

If $\alpha_d \leq \alpha$, then we select both match-keys mk_x and mk_{x+1}. If $\alpha_d > \alpha$ then we stop the selection of candidate match-keys at that point. This process is performed for all statistical tests, resulting in a filtered matrix of plaintext match-keys for each encoded match-key. Table 6.2 shows an example matrix $\mathbf{C}$ obtained after the filtering process

Table 6.2: An example correlation matrix **C** for the encoded match-key (*FirstName* + *LastName* + *Gender*) after the filtering process discussed in the third step of the attack. Crossed-out values represent the plaintext match-keys that are not selected for the corresponding metric due to their low correlation.

Plaintext match-key	Mean	Skewness	....	EMD	Entropy	KS-text
(*FirstName* + *LastName* + *Gender*)	0.985	~~0.541~~	....	0.967	0.901	~~0.389~~
(*FirstName* + *LastName* + *YearOfBirth*)	~~0.662~~	0.877	....	0.989	~~0.578~~	0.998
(*LastName* + *Gender* + *YearOfBirth*)	0.988	0.921	....	0.969	0.974	0.986

for the encoded match-key (*LastName* + *Gender* + *YearOfBirth*). Plaintext match-keys with crossed-out correlation values are not selected for that particular correlation metric due to having a too low correlation value, as discussed in the above filtering step. At the end of this step, we obtain one filtered correlation matrix $\mathbf{C}_j$ per encoded match-key, $\mathbf{H}_{*j}$. This matrix $\mathbf{C}_j$ will contain plaintext match-keys, mk^p, that have the highest correlation values based on the most similar frequency distributions to that encoded match-key $\mathbf{H}_{*j}$.

6.2.4 Reidentifying Encoded Match-Keys

In the fourth step of the attack, we aim to identify the correct plaintext match-key for each encoded match-key, $\mathbf{H}_{*j}$, from its corresponding matrix $\mathbf{C}_j$. In this section, since we are analysing a single encoded match-key at a time, we exclude j from all notations to improve readability. Therefore, it is important to note that the following process is applied for each encoded match-key $\mathbf{H}_{*j}$ separately. For a given filtered matrix **C**, we first calculate an overall correlation, a_c, of each plaintext match-key mk^p by averaging the correlation value vector of that match-key. As we discussed above, this vector consists of the normalised correlation values for the ten different statistical tests. Given that a plaintext match-key $mk^p \in \mathbf{C}$ has a correlation vector $\mathbf{c}_p$, the overall averaged correlation a_c can be calculated as:

$$a_c = \frac{\sum_{i=1}^{10} c_i \in \mathbf{c}_p}{10}. \tag{6.4}$$

Next we look at the number of unique match-key values the plaintext match-key (number of unique concatenated plaintext values) and the encoded match-key (number of unique hash codes) have. Note that even if the distributions of an encoded match-key and a plaintext match-key is similar these two match-keys might have different numbers of unique values. Therefore, apart from correlation measurements, the number of unique values can also be used to compare the two match-keys. If the encoded match-key has u_e unique values and a plaintext match-key $mk^p \in \mathbf{C}$ has u_p unique values, we calculate the difference ratio d_c of those values as:

$$d_c = 2\left(\frac{|u_p - u_e|}{u_p + u_e}\right), \tag{6.5}$$

Table 6.3: An example of plaintext alignment for match-key (*FirstName* + *LastName* + *YearOfBirth*) (F+L+B) showing frequencies of match-key and plaintext values. As discussed in the fifth step of the attack in Section 6.2.5, the encoded match-key value $e1$ from the encoded database $\mathbf{H}$ will be aligned with plaintext match-key value $p1$ from the plaintext database $\mathbf{D}^p$. Since all three encoded match-key values, $e2$, $e3$, and $e4$, have a frequency of 70 and the three plaintext match-key values, $p2$, $p3$, and $p4$, in the plaintext database also have a similar frequency of 68, we assign each of $e2$, $e3$, and $e4$ to all three values of $p2$, $p3$, and $p4$. Similarly, we assign each of $e5$ and d6 to both values of $p5$ and $p6$ since they all have a frequency of 40.

Encoded database $\mathbf{H}$			Plaintext database $\mathbf{D}^p$		
Match-key ID	Match-key value	Frequency	Match-key ID	(F+L+B)	Frequency
$e1$	heYcgrjawf3AVtt	104	$p1$	brittany+nicole+1987	104
$e2$	JbsJvZQ7lucFDcE	70	$p2$	brian+johnson+1968	68
$e3$	NDZ5v3pzT7tvlko	70	$p3$	james+smith+1991	68
$e4$	1XW13iYzExn4KGZ	70	$p4$	ronald+young+1982	68
$e5$	5qIiMWET4suKARu	39	$p5$	ashley+johnson+1975	38
$e6$	b97VABaDFZSg9OM	40	$p6$	johnny+motley+1989	39

where $|\cdot|$ calculates the absolute value. We next calculate the average of the two values, a_c and d_c. This can be calculated as a weighted average, where we assign certain weights to each value. Assuming the weight values for a_c and d_c are w and $1-w$ respectively (with $0 \leq w \leq 1$), the average value s_c is calculated as:

$$s_c = wa_c + (1-w)d_c. \tag{6.6}$$

For each encoded match-key mk^e we obtain $|\mathbf{C}|$ of these average values where $|\mathbf{C}|$ is the number of plaintext match-keys assigned to that encoded match-key. We then sort the plaintext match-keys according to these s_c values and assign the plaintext match-key with the highest s_c value to mk^e. We repeat this step for all encoded match-keys $\mathbf{H}_{*j}$. At the end of this step we obtain a ranked list of plaintext match-keys that are similar to each of the encoded match-key based on their frequency distributions. Based on the superset pruning method suggested by the authors of the MMK approach [163], we can also check for super- or subsets in the identified plaintext match-keys for each encoded match-key $\mathbf{H}_{*j}$. If any of the encoded match-keys, $\mathbf{H}_{*j}$, has already been assigned to a plaintext match-key mk^p_x, then any of the super- or subsets of the mk^p_x cannot be assigned to another encoded match-key, under the assumption that the first assignment of mk^p_x is correct. An example is discussed in Section 6.1.

6.2.5 Reidentification of Plaintext Values

Once the plaintext match-keys for each encoded match-key $\mathbf{H}_{*j}$ are identified, the final step of the attack is to perform plaintext value reidentification by applying a frequency alignment method [41]. For each encoded match-key, $\mathbf{H}_{*j}$, we obtain the frequency distribution of match-key values, and for the identified plaintext match-

key, mk_j^p, of $\mathbf{H}_{*j}$, we obtain the frequency distribution of the corresponding plaintext values. Both these frequency distributions are then sorted in descending order as shown in Table 6.3. First, we adjust these frequency values according to the sizes of the databases $\mathbf{D}^s$ and $\mathbf{D}^p$. We can adjust either the frequencies of match-keys by multiplying each value by $|\mathbf{D}^p|/|\mathbf{D}^s|$, or alternatively we can adjust the frequency values of plaintext values by multiplying each value by $|\mathbf{D}^s|/|\mathbf{D}^p|$. Then we loop over each pair of frequency aligned match-key value and plaintext value, and assign the ith plaintext value to the corresponding ith match-key value. We continue this process until the differences of their frequencies is greater than Δ_t, a user defined threshold. Given the frequencies of the ith match-key value and plaintext value are f_j^i and f_c^i we calculate the frequency difference Δ_i as:

$$\Delta_i = 2\left(\frac{|f_j^i - f_c^i|}{f_j^i + f_c^i}\right). \tag{6.7}$$

If $\Delta_i > \Delta_t$, then we stop the alignment of plaintext values to encoded match-key values (hash codes). Furthermore, if several adjacent frequencies of match-key values and plaintext values in the sorted lists are the same then it will be difficult to determine which plaintext value corresponds to which match-key value. For instance, if $f_j^i = f_j^{i+1} = f_c^i = f_c^{i+1}$, we cannot be certain that the ith match-key value refers to ith plaintext or $(i+1)$th plaintext value. In such scenarios we assign both the ith and $(i+1)$th plaintext values to both the ith and $(i+1)$th match-key values. An example of this process is illustrated in Table 6.3. As shown, records with very similar frequency values are aligned in order to improve the accuracy of the alignment. It is also important to note that this alignment works when the underlying QID value distributions are similar in both $\mathbf{D}^p$ and $\mathbf{D}^s$. In cases where QID value distributions are dissimilar this frequency alignment can potentially lead to incorrect assignments.

Using the above discussed process we can now reidentify the plaintext values from $\mathbf{D}^p$ that corresponds to the most frequent match-key values in each encoded match-key $\mathbf{H}_{*j}$.

6.3 Complexity Analysis

We evaluate the worst-case computational complexity of our attack by analysing the individual steps of the attack discussed in Sections 6.2.1 to 6.2.5. The complexity of calculating frequency distributions of encoded and plaintext match-keys (first step) is $O(|\mathbf{H}| \cdot |\mathbf{mk}^e|)$ and $O(|\mathbf{D}^p| \cdot 2^k)$, respectively, where k is the number of QID attributes in the plaintext database $\mathbf{D}^p$. This is because in this step the frequency distributions of all possible plaintext match-keys and all encoded match-keys are calculated. The complexity of the calculating correlation between frequency distributions (second step) depends on the different correlation tests used. Kolmogorov-Smirnov test, Pearson's correlation coefficient, relative entropy, and histogram intersection have $O(|\mathbf{D}^p| + |\mathbf{H}|)$ complexity for each, Earth mover's distance has $O(|\mathbf{D}^p| \cdot |\mathbf{H}|)$ complexity, and Spearman's rank correlation has $O(|\mathbf{D}^p| + |\mathbf{H}|)$ complexity.

Filtering candidate plaintext match-keys (third step) has $O(|\mathbf{mk}^p| \cdot |\mathbf{mk}^e| \cdot |\mathbf{C}|)$ complexity where $|\mathbf{C}|$ is the number of columns in $\mathbf{C}$. This is because in this step the attack go through each column $\mathbf{C}_{*j}$ (correlation metric) to filter only the highly correlated plaintext match-keys for each encoded match-key. The complexity of reidentifying encoded match-keys (fourth step) is $O(|\mathbf{mk}^p| \cdot |\mathbf{mk}^e|)$ because for each encoded match-key the attack ranks the candidate plaintext match-keys based on the value s_c as discussed in Section 6.2.4. Finally, the complexity of the plaintext reidentification (final step) is $O(|\mathbf{H}| \cdot |\mathbf{mk}^e|)$ because the attack aligns, for each hash value in a encoded match-key, a plaintext match-key value based on their frequencies.

6.4 Experimental Evaluation

We evaluated our proposed attack method on the MMK encoding approach [163] using two real-world databases. We investigated different parameter settings under different attack scenarios to measure the feasibility of our attack method in real-world linkage situations.

6.4.1 Databases and MMK Encoding

We first used a North Carolina Voter Registration (NCVR) database[1], where we used one snapshot collected in December 2020 to create twenty sensitive ($\mathbf{D}^s$) and plaintext ($\mathbf{D}^p$) database pairs. There were five sets of database pairs containing 20,000 (20k), 100,000 (100k), 200,000 (200k), 500,000 (500k), and 1,000,000 (1M) records, where in each set we had four database pairs which had 100%, 75%, 50%, and 0% matching records between them. Our aim was to evaluate how the differences in the number of matching records between databases affects the accuracy of the attack, as well as to analyse the scalability of our attack. We used the same set of NCVR database pairs for the evaluation of the attacks we describe in Chapters 7 and 8.

The second database was the Michigan voter registration (MVR) database[2], where we also used one snapshot collected in September 2016 to create twenty sensitive ($\mathbf{D}^s$) and plaintext ($\mathbf{D}^p$) database pairs. Similar to the NCVR database pairs, MVR database pairs also had 100%, 75%, 50%, and 0% matching records between them and there were database pairs with 20k, 100k, 200k, 500k, and 1M records.

Following the originally proposed match-key encoding [163], we used the following sets of QIDs for the experiments for the two databases NVCR and MVR.

- NCVR: *FirstName*, *LastName*, *StreetAddress*, *City*, and *ZipCode*
- MVR: *FirstName*, *LastName*, *StreetNumber*, *StreetName*, *City*, and *ZipCode*

These QIDs are commonly used to link records across different databases [40, 210]. We selected 15 and 17 different match-keys for the NCVR and MVR databases,

[1] http://dl.ncsbe.gov/data/
[2] http://michiganvoters.info/

respectively, that includes match-keys with length ranging from two to six QIDs. These match-keys cover potential match-keys of different lengths.

6.4.2 Evaluation Criteria

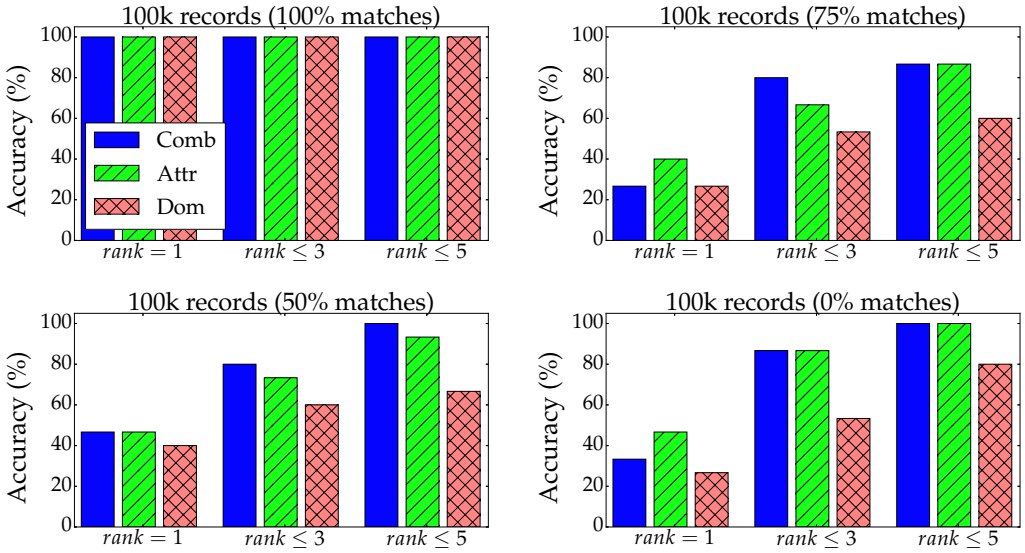
To evaluate the quality of the identification of plaintext match-keys for each encoded match-key $\mathbf{H}_{*j}$, we analysed the top one to five ranked plaintext match-keys assigned for each encoded match-key to see whether the attack was able to identify the correct match-key with a high correlation. We then evaluated the accuracy of the reidentified plaintext QID attribute values for match-key values and reidentified entities using the number of 1-to-1, 1-to-*many*, *many*-to-1, and *many*-to-*many* reidentifications, as we discussed in Chapter 3. We also calculated the probability of suspicion (PS) values for each record in the encoded database $\mathbf{H}$. Furthermore, we defined three different attack scenarios based on the assumed knowledge an adversary has about the match-key encoding settings. The scenarios we considered are as follows:

- **Knowledge about QID attribute combinations (Comb)**: In this scenario we assume that the adversary has knowledge about the actual QID combinations (match-keys) used in the match-key generation process, but does not know which encoded match-key $\mathbf{H}_{*j}$ corresponds to which plaintext match-key $\mathbf{M}_{*j}$. In this scenario the adversary would not have to consider all possible plaintext match-keys. Only the known combinations need to be evaluated to measure the correlation of frequency distributions. This scenario can be considered as the best possible attack scenario on an encoded database from an adversary's point of view.

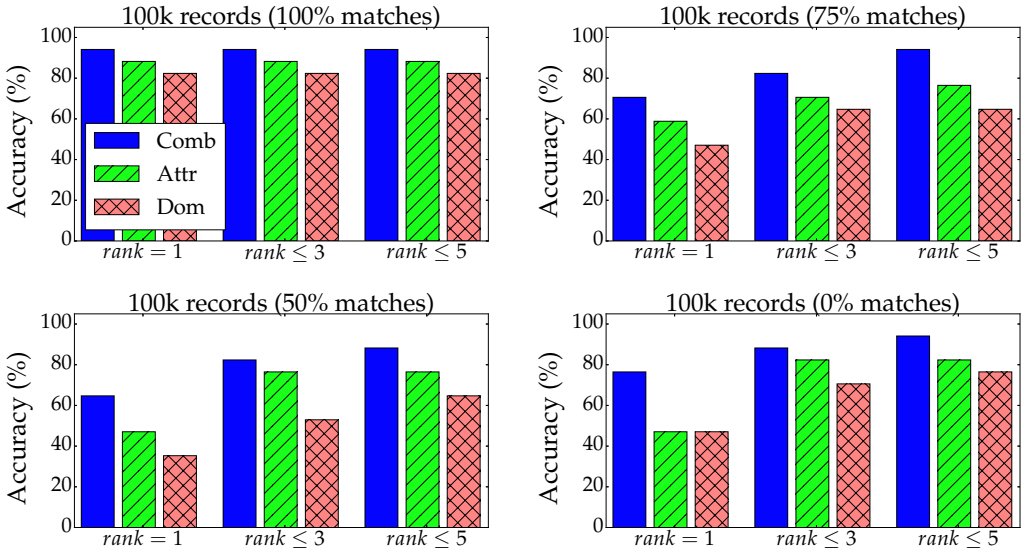
- **Knowledge about used QID attributes (Attr)**: In this scenario we assumed that the adversary has knowledge only about the set of QIDs used for the match-key generation, but does not know which combinations of QIDs are used. Unlike the previous scenario, here the adversary would have to conduct the first step of the attack (as discussed in Section 6.2.1) using all possible plaintext match-keys to identify the best matching plaintext match-keys for each encoded match-key, and then perform the second step of the attack.

- **Domain knowledge only (Dom)**: In this final scenario we assumed that the adversary does not have any specific knowledge about the QIDs, nor the match-keys, used in the match-key generation process. The adversary only knows what kind of entities are encoded in $\mathbf{H}$ (such as people with their names, addresses, and so on) and therefore can guess what kind of QID attributes were being used for the encoding. Hence, the adversary needs to run the attack on all possible match-keys using all QIDs available in $\mathbf{D}^p$ that potentially were encoded into match-keys. This scenario can be considered as the worst-case scenario on an encoded database from an adversary's point of view.

We implemented all attack methods using Python 2.7 and the experiments were run on a server with 64-bit Xeon 2.1 GHz 16-Core CPU, 512 GBytes of memory



(i) Accuracy results for identified match-keys with the NCVR databases.



(ii) Accuracy results for identified match-keys with the MVR databases.

Figure 6.3: Accuracy results for identified match-keys with different databases containing 100%, 75%, 50%, 0% matching record pairs. The results show the percentage of correct plaintext match-keys that were the top, and in the top three and five ranks based on the final correlation value s_c, as we discussed in Section 6.2.4.

and running Ubuntu 18.04. To facilitate reproducibility the prototype programs are available from: https://dmm.anu.edu.au/pprlattack/.

For attack parameter settings, the parameter α can be set to an appropriate value by analysing the correlation values obtained for each correlation metric. If the frequency distributions of the encoded and plaintext match-keys are similar then α can be set to a low value such as $\alpha < 0.1$. Inversely, with contrasting frequency distributions, α can be set to a higher value such as $\alpha = 0.3$ to select a wide range of candidate plaintext match-keys. However, setting α to even a higher value, such as $\alpha > 0.4$, will potentially result in a large selection of candidate plaintext match-keys. We set the weight w used in Equation 6.6 to a higher value than 0.5 because the average correlation value a_c was consistently more accurate in correctly identifying encoded match-keys than the difference ratio d_c in our experiments. The value for Δ_t should be set to a low value such as $\Delta_t < 0.3$ to obtain accurate reidentifications. If the frequency distributions of the two databases are known to have differences, a higher value such as $\Delta_t = 0.4$ can be set. This should however be done carefully because it can lead to inaccurate plaintext reidentifications, because allowing substantial differences in value distributions to be mapped can be wrongly used by the attack to map dissimilar plaintext values with each other. It is also worth noting that the optimal values for these parameters are data dependent and therefore should be set over several iterations of experiments.

In our experimental evaluation, we set $\alpha = 0.05$, $w = 0.7$, $\Delta_t = 0.2$. An adaptive value is selected for the parameter ε based on the maximum frequency value of the considered encoded match-key. A value from 0.1 to 0.9 was first selected and then it is multiplied by the maximum frequency value of the encoded match-key to calculate ε. These settings provided good results in a series of set-up experiments.

6.4.3 Results and Discussion

In Figure 6.3 we show the reidentification results for encoded match-keys used in different databases with different numbers of matching records using the NCVR and MVR databases. As can be seen, with both the databases we were able to identify the QID combinations used for each encoded match-key with high accuracy most of the time. The correct match-key was almost always amongst the top five identified plaintext match-keys. With 100% matching records between the NCVR databases the attack ranked the correct match-key as the top one all the time (i.e. 100% accuracy).

When going from 100% to 0% matching records pairs the accuracy of the identified match-keys drops. This is expected because the frequency distributions of encoded and plaintext match-key values change with less number of common records between the databases. While the results on the databases with 100% matching records illustrate the robustness of the attack in reidentifying encoded match-keys when there is enough frequency information, at the same time the results on the databases with 0% matching records highlight the need of adequate frequency information and distinct frequency distributions for the attack to be successful.

Table 6.4: Attribute reidentification accuracy results for the NCVR and MVR 100k database pairs with different number of matching records. Results are shown for all three considered attack scenarios.

Attack type	Percentage of matches	NCVR / MVR				
		1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
Comb	100%	21 / 26	1 / 1	16 / 20	4 / 13	58 / 78
	75%	10 / 15	0 / 1	3 / 6	2 / 6	56 / 110
	50%	5 / 2	0 / 1	7 / 5	1 / 5	62 / 110
	0%	2 / 4	0 / 1	4 / 5	2 / 4	82 / 131
Attr	100%	21 / 26	1 / 1	16 / 20	4 / 12	58 / 79
	75%	9 / 16	0 / 1	4 / 9	2 / 6	59 / 105
	50%	5 / 6	0 / 1	7 / 4	1 / 4	63 / 103
	0%	2 / 4	0 / 1	5 / 5	2 / 4	83 / 130
Dom	100%	21 / 26	1 / 1	16 / 20	4 / 17	58 / 74
	75%	9 / 8	0 / 1	2 / 6	3 / 6	65 / 108
	50%	5 / 4	0 / 0	7 / 5	1 / 4	62 / 111
	0%	3 / 3	0 / 2	3 / 5	2 / 3	78 / 125

The **Comb** attack scenario provided the best results out of all three attack scenarios. While the **Attr** attack scenario also had good reidentification accuracy, the **Dom** attack scenario provided the worst results. This is because with the **Dom** attack scenario there are a number of different potential candidate match-keys which are sometimes either super- or subsets of the correct match-key. We noticed that when a correct match-key is not ranked as first (top), the match-key that ranked top is always a super- or a subset of the correct match-key. This is because the correlation values measured using the super- and subsets of the correct match-key are highly similar and it is difficult to distinguish the correct match-key from other these possible candidates.

In Tables 6.4 and 6.5 we show the attribute and identity reidentification results for our proposed attack with the NCVR 100k database pairs. As can be seen, the attack was able to reidentify more encoded QID attribute values with high percentages of matches between sensitive and plaintext databases. However, the number of identified QID values drops with lower percentages of matches between two databases. This is because even if the attack was able to identify the correct QID combinations used for encoded match-keys by analysing the frequency distributions, exact reidentifications of plaintext values are difficult as described in Section 6.2.5 due to differences in the actual QID values in records. With different knowledge based attack scenarios the attack maintains fairly consistent results. As expected with the **Dom** scenario the numbers of reidentifications are slightly lower than with the other scenarios because the accuracy of the reidentification of match-keys is lower.

Compared to the attribute reidentification results shown in Table 6.4, the attack produced lower results for identity reidentification, as can be seen in Table 6.5. This is because the attack is based on frequent match-key values, and in the plaintext reidentification step the attack assigns plaintext match-key values to encoded match-

Table 6.5: Identity reidentification accuracy results for the NCVR and MVR 100k database pairs with different number of matching records. Results are shown for all three considered attack scenarios.

Attack type	Percentage of matches	NCVR / MVR				
		1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
Comb	100%	0 / 0	0 / 0	0 / 3	428 / 599	275 / 547
	75%	0 / 0	0 / 0	0 / 0	131 / 218	347 / 825
	50%	0 / 0	0 / 0	0 / 0	97 / 52	474 / 473
	0%	0 / 0	0 / 0	0 / 0	0 / 0	586 / 1150
Attr	100%	0 / 0	0 / 0	0 / 3	428 / 599	275 / 547
	75%	0 / 0	0 / 0	0 / 0	131 / 220	347 / 823
	50%	0 / 0	0 / 0	0 / 0	97 / 109	474 / 933
	0%	0 / 0	0 / 0	0 / 0	0 / 0	586 / 1150
Dom	100%	0 / 0	0 / 0	0 / 3	428 / 601	275 / 545
	75%	0 / 0	0 / 0	0 / 0	128 / 201	350 / 842
	50%	0 / 0	0 / 0	0 / 0	97 / 97	474 / 955
	0%	0 / 0	0 / 0	0 / 0	0 / 0	586 / 1150

key values based on the similarities of their frequencies. For instance, in Table 6.3 the plaintext match-key value 'brittany+nicole+1987' is assigned to the encoded match-key 'heYcgrjawf3AVtt' because both of their frequencies are 104. Even though this can be a unique attribute reidentification, there are 104 entities in each of the plaintext and encoded databases with the same set of values (assuming databases are deduplicated). Therefore, the identity reidentification will be a *many*-to-*many* assignment. Note that for database pairs with 0% matching records, there will not be any correct entity reidentifications because these database pairs do not have entities in common.

Similar results can also be observed in Table 6.6 where we show the PS values for the reidentification of entities. None of the databases and knowledge based attack scenarios had a maximum risk of 1.0, and therefore the marketer risk was always 0. A maximum risk value 0.5 indicates that there were at least one 2-to-2 identity reidentifications where two encoded entities are assigned to two plaintext entities, whereas a minimum risk value 0 indicates that there were records in the databases that had no identity reidentifications. Similar to Table 6.5, database pairs with 0% matches will only have 0 values for the same reason mentioned above.

While these results illustrate the ability of our attack in reidentifying encoded match-keys when there is enough frequency information that can be exploited by the attack, at the same time the results also highlight the difficulty of identifying entities in the encoded database uniquely using only the QID value combinations that are highly frequent.

To analyse the scalability of the attack method, in Table 6.7 we show the runtime and memory consumption results for different steps of our attack. As can be seen, the correlation calculation between encoded and plaintext match-keys (as discussed in Section 6.2.2) is the most time consuming step of the attack requiring a maximum of 2,118 seconds. Calculating frequency distributions and reidentifying plaintext

Table 6.6: Maximum, minimum, mean, median, and marketer risk values of the records in the NCVR and MVR encoded sensitive databases calculated based on probability of suspicion (PS) as we discussed in Section 3.7.1 on page 55.

Attack type	Percentage of matches	NCVR / MVR				
		Maximum risk	Minimum risk	Mean risk	Median risk	Marketer risk
Comb	100%	0.5 / 1.0	0 / 0	0.05 / 0.05	0.04 / 0.03	0 / < 0.001
	75%	0.33 / 0.25	0 / 0	0.04 / 0.04	0.04 / 0.03	0 / 0
	50%	0.06 / 0.07	0 / 0	0.03 / 0.03	0.03 / 0.04	0 / 0
	0%	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Attr	100%	0.5 / 1.0	0 / 0	0.05 / 0.05	0.04 / 0.03	0 / < 0.001
	75%	0.33 / 0.5	0 / 0	0.04 / 0.04	0.04 / 0.03	0 / 0
	50%	0.06 / 0.08	0 / 0	0.03 / 0.02	0.03 / 0.01	0 / 0
	0%	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
Dom	100%	0.5 / 1.0	0 / 0	0.05 / 0.06	0.04 / 0.03	0 / < 0.001
	75%	0.07 / 0.25	0 / 0	0.03 / 0.03	0.04 / 0.02	0 / 0
	50%	0.06 / 0.07	0 / 0	0.03 / 0.02	0.03 / 0.01	0 / 0
	0%	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0

values (as discussed in Sections 6.2.1 and 6.2.5) are the second most time consuming steps which took a maximum of 1,078 and 1,569 seconds, respectively. Selecting and assigning candidate plaintext match-keys for each encoded match-key (as discussed in Sections 6.2.3 and 6.2.4) took less than one second for all the experiments we conducted. Overall the attack took less than 1.5 hours to complete for each of the databases and attack scenarios.

In terms of memory consumption, all steps of the attack had somewhat equal memory usage. When going from 20k to 1M records the memory consumption increased linearly with a maximum memory consumption of around 40GB. The **Dom** attack scenario took the longest time to run and had the largest memory consumption compared to other two scenarios. This is because with the **Dom** a large number of QID attribute combinations (plaintext match-keys) need to be analysed given the adversary does not have any knowledge about the QIDs used for the encoding process. Overall the runtime and memory consumption of the attack are linear in the size of the databases and also depend on the adversary's prior knowledge of the encoded QID attribute values.

Out of all the statistical tests we used to compare frequency distributions, the six correlation metrics Earth mover's distance, Kolmogorov-Smirnov test, Pearson's correlation, Spearman's rank correlation, relative entropy, and histogram intersection provided consistently good results with different attack scenarios and databases, and with different time intervals compared to the four basic statistical measures used. This is due to the ability of correlation metrics to identify common characteristics of distributions even with differences in the actual frequency values.

Table 6.7: Time (in seconds) and memory (in Megabytes) consumption analysis of different steps of the proposed attack. For each step we refer to its corresponding section where we describe that step.

Attack type	Number of records	Time (sec) / Memory (MB) Calculate frequency Section 6.2.1	Calculate correlation Section 6.2.2	Select and assign plaintext match-keys Sections 6.2.3 + 6.2.4	Plaintext reidentification Section 6.2.5
Comb	20k	1 / 4,740	4 / 4,747	0.004 / 4,747	2 / 4,825
	100k	5 / 5,779	15 / 5,800	0.003 / 5,802	14 / 6,239
	200k	11 / 6,963	30 / 7,005	0.003 / 7,012	46 / 7,878
	500k	31 / 10,161	60 / 10,201	0.002 / 10,202	297 / 13,008
	1M	63 / 15,449	125 / 15,485	0.003 / 15,492	1,388 / 21,330
Attr	20k	2 / 4,746	6 / 4,754	0.004 / 4,755	2 / 4,825
	100k	17 / 5,897	46 / 5,914	0.006 / 5,916	27 / 6,358
	200k	15 / 7,029	35 / 7,070	0.003 / 7,075	49 / 7,717
	500k	54 / 10,635	86 / 10,672	0.003 / 10,702	300 / 13,254
	1M	84 / 15,685	119 / 15,707	0.003 / 15,708	1,569 / 19,605
Dom	20k	18 / 5,221	83 / 5,227	0.68 / 5,229	2 / 5,309
	100k	99 / 8,602	273 / 8,609	0.90 / 8,610	18 / 8,982
	200k	207 / 12,540	482 / 12,562	0.62 / 12,566	52 / 13,412
	500k	871 / 23,325	1,516 / 23,369	0.08 / 23,376	512 / 26,064
	1M	1,078 / 40,380	2,118 / 40,452	0.04 / 40,474	1,324 / 46,043

6.4.4 Privacy Improvements and Recommendations

Considering the identified privacy vulnerabilities in the match-key generation process, we now propose two recommendations to strengthen the privacy guarantees of the MMK encoding approach. We further show empirical evidence to support our claim of using these recommendations to make the MMK encoding resilient to frequency based privacy attacks.

Recommendation 1: As we discussed in Section 6.1, encoded match-keys are stored as lists so that only corresponding match-key values are compared when matching two records. For instance, if records are encoded using four match-key values, [mk^e_1, mk^e_2, mk^e_3, mk^e_4], when a record pair $r1$ and $r2$ is compared, $r1[mk^e_1]$ is only compared with $r2[mk^e_1]$, and $r1[mk^e_2]$ is only compared with $r2[mk^e_2]$, and so on. This is because the match-key values are stored as a list in the same order for all records.

This ordering of match-key values allows the calculation of frequency distributions of each match-key in the attack. An adversary can safely assume that a single column in the encoded database $\mathbf{D}^s$ (a column $\mathbf{H}_{*j} \in \mathbf{H}$) corresponds to a single encoded match-key. To prevent such assumptions by the adversary we propose to store match-key values in sets as opposed to lists. Each row $\mathbf{H}_{i*} \in \mathbf{H}$ in the encoded database should be a set of match-key values where there is no order. Now the adversary cannot assume a single column represents a certain encoded match-key.

As opposed to improved privacy, there is a trade-off with this technique. A single match-key value from a record now needs to be compared with every match-key

Table 6.8: Comparison of precision and recall of the linkage when improvement method proposed in recommendation 2 (on page 124) is applied. Maximum difference percentage in both precision and recall for each row is also presented. Different database pairs from both NCVR and MVR are used for the evaluation. Results are shown for different x values where match-key values are set to missing of those values have a frequency larger than x.

Database pair	Number of records	Original Precision/Recall (%)	Precision/Recall after applying recommendation 2 $x = 20$ (%)	$x = 10$ (%)	$x = 5$ (%)	$x = 2$ (%)	$x = 1$ (%)	Maximum difference percentage
NCVR 100% matches	20k	99.8/100	99.8/100	99.8/100	99.8/100	99.8/100	100/99.9	+0.23/-0.01
	100k	98.9/100	98.9/100	98.9/100	98.9/100	98.9/100	100/99.9	+1.12/-0.02
	200k	97.7/100	97.7/100	97.7/100	97.7/100	97.8/100	100/99.9	+2.28/-0.06
	500k	94.4/100	94.4/100	94.4/100	94.5/100	94.8/100	100/99.8	+5.61/-0.11
	1M	89.6/100	89.6/100	89.6/100	89.7/100	90.8/99.9	100/99.8	+10.45/-0.2
NCVR 75% matches	20k	99.7/100	99.7/100	99.7/100	99.3/100	99.7/100	99.9/99.9	+0.24/-0.01
	100k	98.5/100	98.5/100	98.5/100	98.5/100	98.6/100	99.9/99.9	+1.39/-0.03
	200k	96.9/100	96.9/100	96.9/100	96.9/100	97.1/100	99.8/99.9	+2.83/-0.05
	500k	92.8/100	92.8/100	92.8/100	92.8/100	93.3/99.9	99.5/99.8	+6.74/-0.13
	1M	86.6/100	86.6/100	86.6/100	86.8/100	88.2/99.9	99.2/99.7	+12.55/-0.25
MVR 100% matches	20k	99.7/100	99.7/100	99.7/100	99.7/100	99.7/100	100/100	+0.29/0
	100k	98.8/99.9	98.8/99.9	98.8/99.9	98.7/99.9	98.9/99.9	100/99.9	+1.13/0
	200k	97.8/99.9	97.8/99.9	97.8/99.9	97.8/99.9	97.8/99.9	100/99.9	+2.21/-0.03
	500k	94.6/99.9	94.6/99.9	94.6/99.9	94.6/99.9	94.9/99.9	100/99.9	+5.38/-0.08
	1M	89.8/99.9	89.8/99.9	89.8/99.9	89.8/99.9	90.8/99.9	100/99.8	+10.18/-0.14
MVR 75% matches	20k	99.6/99.9	99.6/99.9	99.6/99.9	99.6/99.9	99.6/99.9	99.9/99.9	+0.36/-0.01
	100k	98.6/99.9	98.6/99.9	98.6/99.9	98.6/99.9	98.6/99.9	99.8/99.9	+1.33/-0.01
	200k	97.2/99.9	97.2/99.9	97.2/99.9	97.2/99.9	97.2/99.9	99.8/99.9	+2.66/-0.02
	500k	93.0/99.9	93.0/99.9	93.0/99.9	93.0/99.9	93.4/99.9	99.6/99.8	+6.58/-0.10
	1M	86.9/99.9	86.9/99.9	86.9/99.9	86.9/99.9	88.3/99.9	99.2/99.8	+12.3/-0.18

value from another record which will lead to increased time consumption. Furthermore, the possibility of obtaining false positives will also increase. For instance if two record $r1$ and $r2$ have the following QID values: $r1$: *FirstName* = 'Paul', *LastName* = 'Thomas', *YearOfBirth* = '1992' and $r2$: *FirstName* = 'Thomas', *LastName* = 'Paul', *YearOfBirth* = '1992', then match-key (F+B) of $r1$ and match-key (L+B) of $r2$ will have the same value and the record pair ($r1$, $r2$) will be classified as a match. Increasing false positives will reduce the overall quality of the linkage. However, this can be solved by prefixing an QID specific value (such as an QID qualifier or an QID index) to the plaintext value before hashing.

For example if we use 'F' for first name and 'L' for last name, we obtain match-key values such as 'F-Thomas1992' and 'L-Thomas1992'. This approach is different from record specific salting values as used in the context of Bloom filter encoding to avoid frequency attacks [40]. Therefore, adding an QID specific string may not affect the frequency distribution, rather it is recommended to avoid any false positive matches.

Recommendation 2: As discussed in Section 6.2, our proposed attack compares the correlations between the frequency distributions of encoded match-keys with the frequency distributions plaintext match-keys. In order for this to be successful, the encoded match-keys need to have distinct frequencies for different match-key values.

If the frequency distributions are close to uniform, an adversary will not be able to compare those distributions with plaintext frequency distributions.

As our second recommendation, we propose making the frequency distributions of encoded match-keys close or equal to uniform. In the encoded database $\mathbf{H}$, we set the match-key values that have frequency larger than x, where $x \geq 1$, as missing. This will ensure that most of the remaining match-key values (hash codes) for each encoded match-key are unique (all will be unique if $x = 1$) and the frequencies of all those match-key values will be between 1 and x, making the distribution close to uniform. An adversary can no longer conduct any correlation analysis on these distributions since they are uniform and cannot be distinguished from each other.

As with our first recommendation, this method also has a trade-off for improved privacy at the cost of reduced linkage quality. Setting match-key values as missing will lower the overall recall of the linkage since candidate true matches could be missed. At the same time, overall precision of the linkage will be improved since candidate false positives could be removed when using this technique.

We conducted multiple experiments with x = [20, 10, 5, 2, 1] using both the NCVR and MVR databases where we set the match-key values with frequency larger than x as missing. In Table 6.8 we present the results of those experiments. As expected, the precision of the linkage increased with a minimum and a maximum increase of +0.23% and +12.3%, respectively. The recall dropped due to the true matches being missed because their match-key values are removed. The overall recall had a minimum and a maximum decrease of -0.01% and -0.25% respectively. The small changes in both precision and recall is because the number of records with match-key value frequencies larger than 1 is small compared to the actual number of records in the databases. However, if the databases have more errors or missing values in QIDs this will potentially lead to a selection of encoded match-keys, $\mathbf{H}_{*j}$, with a small number of QIDs which will improve the recall of the linkage. As we further discuss below, selecting encoded match-keys with a smaller number of QID values (such as only two or three QIDs in a $\mathbf{H}_{*j}$) will increase the frequencies of those encoded match-keys. In such a case using this recommendation to improve privacy will also lead to a potentially substantial reduction of recall.

It is possible to apply one or both of the above two recommendations to increase the privacy of the sensitive values in the encoded database. However, we recommend to initially apply the first recommendation and then check if any distinct frequency distributions can still be obtained and aligned between the plaintext and encoded databases. Depending on the results of the first modification, we can then apply the second recommendation if further improvement of privacy is required. Furthermore, it is also worth noting that if the second recommendation is applied first, then there is no practical advantage of applying the first recommendation next because the attack will not be possible after the application of the second recommendation.

6.5 Practical Considerations of the MMK Encoding

While the original MMK encoding approach proposed by Randall et al. [163] provides high linkage quality with acceptable privacy, there are few practical aspects that need careful consideration when using this encoding in real-world applications.

1. The selection of match-keys depends on the Fellegi and Sunter agreement and disagreement weights of individual QIDs. The authors have conducted their experiments on selected match-keys that provided the highest F-measure value for a given database [163]. However, in practice these agreement and disagreement weights need to be estimated based on (partially) available ground truth or domain knowledge. Hence, with estimated values, the calculated F-measure will not be the best and thus the final record linkage quality will not be optimal.

2. The MMK encoding approach depends on exact matching of hash codes. As also discussed by the authors in [163], this could be a disadvantage if the matching databases contain errors or variations in QID values. Even a small typographical error or a character change (Christine vs. Christina) in a QID value will lead to a completely different hash code, whereas with approximate matching techniques such as Bloom filter encoding [177] this might not lead to false negatives. Furthermore, in our experiments, we observed that the match-keys selected using F-measure always contain a name QID attribute (*FirstName*, *MiddleName*, or *LastName*). This is because name QID values generally have more discriminatory power compared to other QID attributes used. Therefore, if the name QIDs in the databases to be linked contain more errors or changes in their values (over time), then this could lead to lower linkage quality despite low amounts of errors in other used QID attributes.

3. When the sizes of the databases get larger, it is more likely for them to have a distinct frequency distribution for each match-key (QID attribute combination). Even if the authors of the MMK encoding approach [163] suggested to have two or more QID attribute values in a match-key, the results of our experiments showed that having even three QID attribute values in a match-key will still lead to a unique frequency distribution for that particular match-key which can lead to reidentifications. In our experiments on databases with different number of records, the frequencies start to get close to uniform with match-keys made of three or more QID attributes. But then again, if we are using five or more QIDs for a match-key we need to make sure these QID attributes do not contain a large number of errors or missing values because that will lead to lower linkage quality.

6.6 Summary

We have proposed a privacy attack on the recently proposed multiple match-key (MMK) encoding technique for PPRL [163]. The proposed attack employs correlation

calculations on frequency distributions to identify QID attribute combinations used to generate match-key values. Using the identified match-keys, the attack then reidentifies QID values that corresponds to individual match-key values (hash codes). The experimental results showed that MMK encoding is susceptible to frequency attacks in situations where the adversary has access to a plaintext database with similar match-key frequency distributions to the encoded database. As countermeasures, we have proposed two countermeasures to improve the privacy of the MMK encoding approach and have discussed the trade-offs between linkage quality and improved privacy when using these methods. These recommendations show that with a small number of errors and missing values in databases, privacy can be strengthened while keeping high linkage quality.

Chapter 7

Pattern-mining based Cryptanalysis on Bloom Filter Encoding

In this chapter we propose a pattern-mining based attack method on Bloom filter encoding for privacy-preserving record linkage (PPRL). In this attack, using a set of refinement steps, we extend a previously proposed frequency based attack on PPRL [43] to substantially improve the reidentification results.

In Section 7.1 we first provide a brief introduction to Bloom filter encoding and our attack method, and in Section 7.2 we then detail our contributions. In Section 7.3 we provide an overview of our attack and in Sections 7.4 to 7.7 we describe the main steps of our attack including the expansion steps to improve the reidentification results. In Section 7.8 we analyse the computational complexity of our attack and in Section 7.9 we then experimentally evaluate our proposed attack method using a real-world database. Finally, in Section 7.10 we provide a summary of our findings.

7.1 Introduction

As we discussed in Chapter 2 one widely used perturbation technique for privacy-preserving record linkage (PPRL) is Bloom filter (BF) encoding [108, 177]. A BF is a bit array where all bit positions are initialised to 0. The elements of a set are hash mapped into a BF using multiple hash functions that set certain bit positions in the BF to 1 [24]. In PPRL, these elements are commonly q-grams of length l_q extracted from the string values in records that are being encoded. The number of common 1-bits between two BFs approximates the number of common q-grams between two string values allowing the calculation of the similarity between two BFs [177]. A more detailed and formal description of BF encoding is given in Appendix A.

Due to its ability to allow accurate and efficient linkage, as well as easy implementation, BF encoding is now being used in real-world PPRL systems across the world [10, 29, 153, 154, 164]. However, recent research has shown that basic BFs are vulnerable to certain types of attacks as we described in Chapter 4 [41, 42, 119, 121, 139, 147]. These attacks mainly exploit the frequencies of bit patterns in a set of BFs, and how these correlate to frequent plaintext values in a plaintext database. This

Table 7.1: Specific notation used throughout this chapter. For common notation used throughout the thesis, refer to the list of notation at the beginning of the thesis.

$\mathbf{B}$	List of Bloom filters from $\mathbf{D}^e$
$\mathbf{V}$	List of plaintext quasi-identifying (QID) attribute values from $\mathbf{D}^p$
$\mathbf{F}$, $\mathbf{F}_E$, $\mathbf{F}_F$	Lists of identified, expanded set of, and final set of frequent q-grams and their bit positions
$\mathbf{A}^+$, $\mathbf{A}_E^+$, $\mathbf{A}_F^+$	Lists of identified, expanded, and final must have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{A}^-$, $\mathbf{A}_E^-$, $\mathbf{A}_F^-$	Lists of of identified, expanded, and final cannot have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{R}$	List of reidentified plaintext QID values for each Bloom filter
$\mathbf{B}_c$	Column-wise storage of BF list $\mathbf{B}$
$\mathbf{q}$	Set of q-grams extracted from a QID value
$\mathbf{z}$	Sequence of indicator variables to calculate the probability of collision in BFs when hashing a single element
f	Frequency of a q-gram in the list $\mathbf{Q}$
b	An index of a Bloom filter $\mathbf{b}$
$v \in \mathbf{V}$	A plaintext QID value
k	Number of hash functions
k_{est}	Estimated number of hash functions used to encode q-grams
w_H	Hamming weight of a BF
d	Minimum percentage difference between two most frequent q-grams
m	Minimum partition size (a subset of BFs in $\mathbf{B}$) as a number of BFs
n_m	Minimum number of required identified q-grams in a BF from $\mathbf{B}$

allows an adversary to reidentify the encoded quasi-identifying (QID) attribute values in BFs. Nevertheless, most existing attacks on BFs require knowledge of certain parameters used in the BF encoding process, and/or have high computational costs making them impractical for attacking large databases.

In this chapter we describe a novel cryptanalysis attack which overcomes the drawbacks of existing attacks. Apart from the length of q-grams, l_q, and the types of QID attributes that have been encoded into BFs, our attack does not require any knowledge of the parameters used in the BF encoding process, and it is more effective in reidentifying encoded values compared to most existing attacks. Most importantly, our attack can be successful even if each BF in an encoded database is unique, which makes the attack more practical in real-world scenarios.

7.2 Contribution

The initial version of this attack method was proposed by Christen et al. [43] where the authors only presented the basic steps of the attack. This chapter presents a significantly improved and extended attack method compared to this previous work.

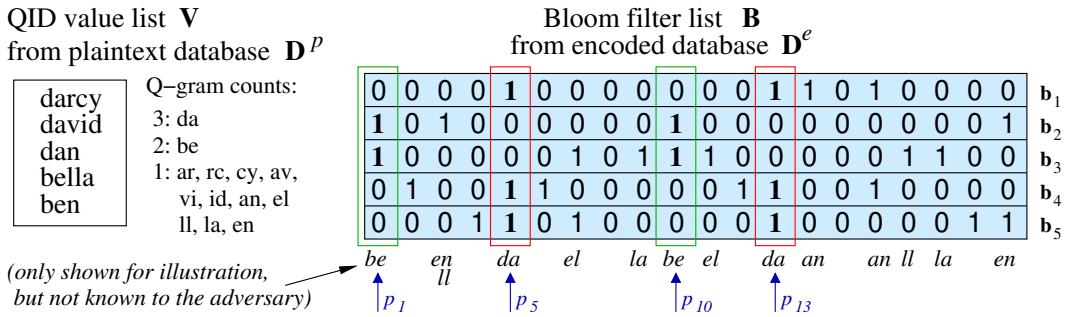


Figure 7.1: An example illustration of the basic idea of our proposed attack method, as described in detail in Section 7.3. Using a pattern-mining approach, we first identify that bit positions p_5 and p_{13} have co-occurring 1-bits in the same three BFs ($\mathbf{b}_1$, $\mathbf{b}_4$ and $\mathbf{b}_5$), and therefore must encode 'da' which is the only q-gram that occurs in three plaintext values. Next, we find that positions p_1 and p_{10} must encode 'be' because they have co-occurring 1-bits in the same two BFs ($\mathbf{b}_2$ and $\mathbf{b}_3$) and 'be' is the only q-gram that occurs in two plaintext QID values. An adversary now knows that BFs $\mathbf{b}_2$ and $\mathbf{b}_3$ can only encode 'bella' or 'ben' while $\mathbf{b}_1$, $\mathbf{b}_4$, and $\mathbf{b}_5$ can encode either 'darcy', 'david', or 'dan'.

Our attack is based on a maximal frequent itemsets pattern-mining approach. As illustrated in Figure 7.1, we first identify frequently co-occurring bit positions in a BF database that encode frequent q-grams in a plaintext database. Using a frequent q-gram based probabilistic language model and a partitioning method we then expand the set of identified q-grams. The language model identifies subsets of BFs that are known to encode a certain previously identified frequent q-gram, allowing us to identify new q-grams that co-occur with this already identified frequent q-gram with high probability. In the partitioning method we divide the BF and plaintext databases into pairs of subsets using different combinations of frequent q-grams, and then apply pattern-mining on those subsets to identify further q-grams and their co-occurring bit positions. Based on these identified q-grams and their corresponding bit positions we then try to reidentify plaintext QID values encoded in BFs.

We theoretically analyse the applicability of frequent pattern-mining in our attack to correctly identify the co-occurring bit positions in BFs that encode a frequent q-gram. In order to find inaccurate bit positions assigned to the q-grams identified in the first step of our attack we propose a constraints based filtering approach. We theoretically analyse and justify the applicability of the probabilistic language model as used in the second step of our attack, and we use a length filtering approach to improve the accuracy of the reidentified plaintext values. Finally, we evaluate our attack on several real databases with different characteristics. These experiments show that our attack can identify bit positions of q-grams with high precision, and it can reidentify plaintext values even when each BF in an encoded database is unique. In such a scenario no previous attack on BF encoding for PPRL would have been successful.

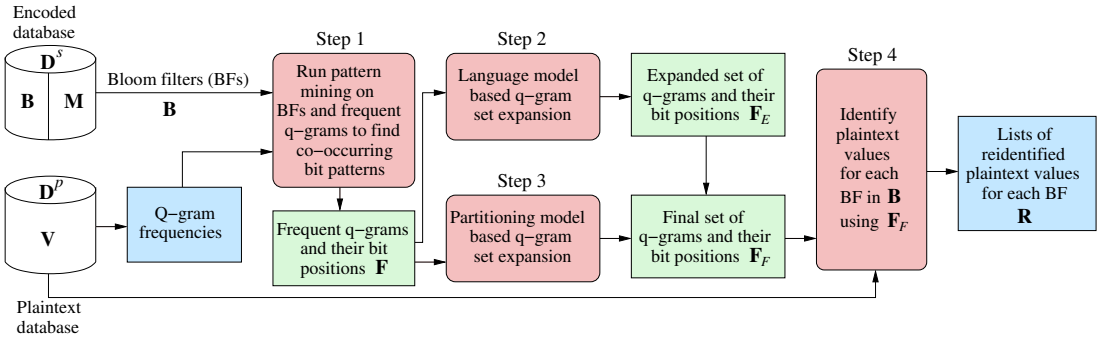


Figure 7.2: An overview of our proposed attack method. The adversary has access to the encoded BF database and a plaintext database from which he extracts q-gram frequencies. The first three steps the attack returns three sets of q-grams and their identified bit positions $\mathbf{F}$, $\mathbf{F}_E$, and $\mathbf{F}_F$. In the final step, using those identified q-grams the attack reidentifies possible plaintext values encoded in each BF. The list of identified plaintext values for each BF is returned as $\mathbf{R}$. The initial inputs and the final output of the attack are shown in blue while intermediate inputs and outputs are shown in green.

In the following we discuss each step of our attack in more detail. Specific notation we use throughout this chapter are shown in Table 7.1.

7.3 Overview of the Bit Pattern based Cryptanalysis Attack

In this section we first discuss the main assumptions of our attack, and then we show the probability of getting a certain Hamming weight (number of 1-bits in a bit vector) when hashing a single element into a BF. We also discuss the applicability of different pattern-mining algorithms for our attack.

7.3.1 Preliminaries and Assumptions

As with existing attacks on BFs for PPRL, we assume the adversary has access to an encoded database, $\mathbf{D}^e = (\mathbf{B}, \mathbf{M})$, where $\mathbf{B}$ is the list of BFs and $\mathbf{M}$ is the list of microdata for each record. The adversary does not know which plaintext QID value(s) from a sensitive database, $\mathbf{D}^s$, are encoded in $\mathbf{D}^e$. The adversary also has access to a plaintext database, $\mathbf{D}^p = (\mathbf{V})$, where $\mathbf{V}$ is the list of values from one or several QID attributes that are from the same domain as the QID attributes encoded in $\mathbf{D}^e$. Similar to most existing attacks, we assume that the adversary can guess which attributes are encoded in $\mathbf{B}$ and what value was used for the length of a q-gram (l_q) based on the distribution of 1-bits in the BFs in $\mathbf{D}^e$ [180]. However, unlike with the other attacks discussed in Chapter 4, our attack does not require any knowledge about any other parameters used in the BF encoding process.

Our attack exploits the BF construction principle [24] as follows in order to identify sets of frequently co-occurring bit positions in $\mathbf{B}$ that can encode frequent q-grams of QID values in $\mathbf{V}$:

Proposition 7.3.1. *Assume a sensitive plaintext database (unknown to the adversary),* $\mathbf{D}^s = (\mathbf{Q}^s, \mathbf{M})$*, that contains* $n = |\mathbf{D}^s|$ *records, and* $k \geq 1$ *independent hash functions are used to encode q-grams extracted from QID values* $v^s \in \mathbf{Q}^s$ *into the encoded database* $\mathbf{D}^e = (\mathbf{B}, \mathbf{M})$ *of n BFs. Assume a q-gram q occurs in* $n_q < n$ *records in* $\mathbf{D}^s$*. Then (1) each bit position (column) in* $\mathbf{B}$ *that can encode q must contain a 1-bit in at least* n_q *BFs in* $\mathbf{B}$*, and (2) if* $k > 1$ *then up to k bit positions must contain 1-bits in the same subset of* n_q *BFs in* $\mathbf{B}$ *that encode the q-gram q.*

Proof. For (1), based on the BF construction principle [24], if a BF contains a 0-bit in a bit position p where any of the k hash functions have mapped a q-gram q into, then this BF cannot encode q. Formally, $\exists\ h_j, 1 \leq j \leq k : (h_j(q) = p) \wedge (\mathbf{b}[p] = 0) \Rightarrow \mathbf{b}$ cannot encode q. For (2), given two distinct BF bit positions, p_x and p_y (with $p_x \neq p_y$), for a given BF $\mathbf{b}$, if $\mathbf{b}[p_x] \neq \mathbf{b}[p_y]$ then $\mathbf{b}$ cannot encode a q-gram q because $\forall h_i, h_j, 1 \leq i, j \leq k, i \neq j : (h_i(q) = p_x \wedge h_j(q) = p_y) \Rightarrow (\mathbf{b}[p_x] = 1) \wedge (\mathbf{b}[p_y] = 1)$. □

In reality, an adversary will unlikely have access to a plaintext database that contains exactly the same number of records as the encoded database. Any assumption about matching frequencies between q-grams in $\mathbf{V}$ and 1-bit patterns in $\mathbf{B}$ therefore needs to be adjusted, as we describe in Section 7.4.

Following Proposition 7.3.1, our pattern-mining based attack relies on a set of bit positions that have frequently co-occurring 1-bit patterns. However, since there can be collisions of the k hash functions when a q-gram is hashed into BFs [140], potentially less than k bit positions will encode a certain q-gram. Therefore less than k bit positions might be co-occurring frequently for a certain frequent q-gram.

7.3.2 Probability of Collision When Hashing One Element into a Bloom Filter

We can calculate the probability of obtaining a certain Hamming weight (number of 1-bits) when a q-gram is hashed k times as follows. Assuming the length of a BF as l, we can use the birthday paradox [140] to calculate the probability that hashing a single q-gram q using k independent hash functions will lead to a Hamming weight w_H of a (initialised to all 0-bits) BF, with $1 \leq w_H \leq k$. We consider all possible combinations for each of the k hash functions to either hash q to a 0-bit (a new position, i.e. no collision) or a 1-bit (a collision, where a previous hash function has already mapped q into). Let us denote with $\mathbf{z} = [z_1, \ldots, z_k]$ the sequence of indicator variables with $z_i \in \{0, 1\}$, $1 \leq i \leq k$, where $z_i = 0$ means a hash function h_i maps the q-gram q to a 0-bit and $z_i = 1$ means h_i maps q to a 1-bit. The probability of a given sequence $\mathbf{z}$ occurring can then be calculated as:

Table 7.2: Probabilities of obtaining a certain Hamming weight, w_H, when hashing a single q-gram k times into a BF of length l =1,000 for different k. Probabilities are rounded to four decimal places.

	$\rho(w_H$=k)	$\rho(w_H$=k-1)	$\rho(w_H$=k-2)	$\rho(w_H$=k-3)	$\rho(w_H$=k-4)
Collisions	0	1	2	3	4
$k = 10$	0.9558	0.0434	0.0007	< 0.0001	< 0.0001
$k = 15$	0.8999	0.0958	0.0042	< 0.0001	< 0.0001
$k = 30$	0.6445	0.2887	0.0589	0.0073	0.0006
$k = 50$	0.2877	0.3706	0.2258	0.0866	0.0235

$$\rho(\mathbf{z}) = 1 \cdot \prod_{i=2}^{k} \left[\left(\frac{i - \sum_{j=1}^{i} z_j}{l} \right)^{z_i} \cdot \left(\frac{l - \sum_{j=1}^{i-1} (1 - z_j)}{l} \right)^{1-z_i} \right]. \tag{7.1}$$

The 1 at the beginning of Equation 7.1 means the first hash function h_1 always maps a q-gram to a 0-bit (assuming a BF is initialised to all 0), and the product therefore starts from $i = 2$. The first factor in the product corresponds to a collision (h_i maps to a 1-bit) and the second to a non-collision (h_i maps to a 0-bit), which is one factor of the birthday paradox.

In total, there will be 2^{k-1} different sequences of $\mathbf{z}$ (because always $z_1 = 0$). For a given sequence $\mathbf{z}$, the Hamming weight of the resulting BF will be the number of hash functions that map q-gram q to a 0-bit, $\omega_H(\mathbf{z}) = |\{\mathbf{z} : z_i = 0, 1 \leq i \leq k\}|$. The overall probability that hashing a single q-gram using k independent hash functions will result in a Hamming weight of w_H, with $1 \leq w_H \leq k$, is therefore $\rho(w_H) = \sum_{\mathbf{z} \in \mathbf{Z}: \omega_H(\mathbf{z}) = w_H} \rho(\mathbf{z})$, where $\mathbf{Z}$ is the set of all 2^{k-1} possible sequences of z_i's of length k where $z_1 = 0$.

Table 7.2 shows probabilities $\rho(w_H)$ for the commonly used BF length l = 1,000 and up-to $k = 50$ hash functions [61, 177]. As can be seen, more than one collision ($w_H < k - 1$) for the same q-gram is unlikely. For example, setting $k = 30$ [180] leads to a probability of 0.64 of no collision for a single q-gram, and a probability of less than 0.067 of more than one collision. For $k = 20$ the probability of no collision is 0.83 while for $k = 10$ it is 0.96. Therefore, our attack should be able to identify k (or a few less than k) frequently co-occurring bit positions that encode the same q-gram, as we experimentally validate in Section 7.9.

7.3.3 Applicability of Pattern-mining Techniques

We now describe the applicability of pattern-mining algorithms in the context of our attack. The idea of our pattern-mining based attack method, as illustrated in Figure 7.1, is to identify a set of BF bit positions that encode a single q-gram. In the context of pattern-mining, BF bit positions would be considered as items, while BFs correspond to itemsets. Each 1-bit means the corresponding position is part of an itemset. For example, the itemset for BF $\mathbf{b}_1$ in Figure 7.1 is $\{p_5, p_{13}, p_{14}, p_{16}\}$.

Algorithm 7.1: Identify frequent q-grams and co-occurring BF bit positions

Input : $\mathbf{B}$: List of BFs from the sensitive encoded database $\mathbf{D}^e$
$\mathbf{V}$: List of plaintext QID values from a public database $\mathbf{D}^p$
l_q: Length of substrings to extract from plaintext values
d: Minimum percentage difference between two most frequent q-grams
m: Minimum partition size (a subset of BFs in $\mathbf{B}$) as a number of BFs

Output: $\mathbf{F}$: List of identified frequent q-grams with their corresponding BF bit positions
$\mathbf{A}^+$: List of must have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{A}^-$: List of cannot have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{Q}$: List of q-gram sets, one per plaintext value in $\mathbf{V}$
$\mathbf{B}_c$: Column-wise storage of BF list $\mathbf{B}$
k_{est}: Estimated number of hash functions used to encode q-grams

$\mathbf{B}_c \leftarrow ConvColWise(\mathbf{B})$ // Generate one bit array per BF bit position (column)
$l \leftarrow |\mathbf{B}_c|$ // Number of BF bit positions (BF length)
$\mathbf{Q} \leftarrow GenQGramSets(\mathbf{V}, l_q)$ // Convert plaintext values into q-gram sets
$\mathbf{F} \leftarrow [\,]$, $\mathbf{A}^+ \leftarrow [\,]$, $\mathbf{A}^- \leftarrow [\,]$, $\mathbf{m} \leftarrow \{\}$, $\mathbf{n} \leftarrow \{\}$, $\mathbf{k} \leftarrow [\,]$ // Initialise lists to be used
$\mathbf{r} \leftarrow \{b : 1 \leq b \leq |\mathbf{B}|\}$, $\mathbf{c} \leftarrow \{p : 1 \leq p \leq l\}$ // Initialise the column filter
$Q \leftarrow [(|\mathbf{B}|, \mathbf{c}, \mathbf{r}, \mathbf{m}, \mathbf{n})]$ // Initialise queue with the first partition (full BF database)
while $|Q| > 0$ **do**: // Main loop: run until the queue is empty
  $(ps_i, \mathbf{c}_i, \mathbf{r}_i, \mathbf{m}_i, \mathbf{n}_i) \leftarrow Q.pop()$ // Start iteration i: Get the first tuple from queue
  $q_1, f_1, q_2, f_2 \leftarrow GetTwoMostFreqQGrams(\mathbf{Q}, \mathbf{m}_i, \mathbf{n}_i)$ // In current partition
  $d_i \leftarrow 2(f_1 - f_2)/(f_1 + f_2) \cdot 100$ // Percentage difference between q_1 and q_2
  if $d_i \geq d$ **then**: // Only continue if percentage difference is large enough
    $s_i \leftarrow |\mathbf{B}| \cdot (f_1 + f_2)/(2|\mathbf{V}|)$ // Minimum support for frequent pattern-mining
    $\mathbf{f}_i \leftarrow GetMaxFreqCoOccurBitPos(\mathbf{B}_c, s_i, \mathbf{c}_i, \mathbf{r}_i)$ // Run pattern-mining
    if $|\mathbf{f}_i| > 0$ **then**: // A set of frequent co-occurring bit positions is identified
      $\mathbf{F}.append((q_1, \mathbf{f}_i))$ // Assign bit positions to most frequent q-gram q_1
      $\mathbf{k}.append(|\mathbf{f}_i|)$ // Assign the length of bit positions as estimated k
      $\mathbf{c}_i \leftarrow \mathbf{c}_i \setminus \mathbf{f}_i$ // Mark identified bit positions as assigned
      $\mathbf{r}_i^+, \mathbf{r}_i^- \leftarrow UpdateRowFilter(\mathbf{f}_i, \mathbf{r}_i)$ // Based on bit positions in $\mathbf{f}_i$
      $\forall b \in \mathbf{r}^+ : \mathbf{A}^+[b] \leftarrow \mathbf{A}^+[b] \cup \{q_1\}$ // Assign q_1 to must have q-gram sets
      $\forall b \in \mathbf{r}^- : \mathbf{A}^-[b] \leftarrow \mathbf{A}^-[b] \cup \{q_1\}$ // Assign q_1 to cannot have q-gram sets
      if $|\mathbf{r}_i^+| \geq m$ **then**: // Partition containing q_1 is large enough
        $Q.append((|\mathbf{r}_i^+|, \mathbf{c}_i, \mathbf{r}_i^+, \mathbf{m}_i \cup \{q_1\}, \mathbf{n}_i))$ // New partition of BFs
      if $|\mathbf{r}_i^-| \geq m$ **then**: // Partition not containing q_1 is large enough
        $Q.append((|\mathbf{r}_i^-|, \mathbf{c}_i, \mathbf{r}_i^-, \mathbf{m}_i, \mathbf{n}_i \cup \{q_1\}))$ // New partition of BFs
      $Q.sort_largest_first()$ // Sort queue by partition size with the largest first
$k_{est} \leftarrow mode(\mathbf{k})$ // Assign most occurred as the estimated k
return $\mathbf{F}$, $\mathbf{A}^+$, $\mathbf{A}^-$, $\mathbf{Q}$, $\mathbf{B}_c$, k_{est}

The way BFs are generated leads to some specific aspects of how pattern-mining can be applied: (1) We assume each BF in an encoded database is unique and therefore each itemset is unique, and (2) we are only interested in the maximal (longest) frequent itemsets which have an expected length of k, the number of hash functions used to encode q-grams into BFs. As a result, four pattern-mining algorithms were considered for our attack.

Apriori [4] is a breadth-first search approach which iteratively generates candidate itemsets of increasing length based on the principle that if an itemset is frequent then all of its subsets must also be frequent, and vise versa. Apriori requires one

database scan in each iteration and it has to store all candidate itemsets of a certain size. **FPmax** [81] extends the FP-Growth (Frequent Pattern Growth) algorithm to only mines the maximal frequent itemsets from an FP-Tree. The algorithm builds a maximal frequent itemset tree to keep track of all maximal frequent itemsets which makes traversing the FP-tree faster.

Max-Miner [18] algorithm orders the candidate sets into a set-enumeration tree and employs a breadth-first search over this tree to limit the total number of scans of the database. The algorithm utilises subset infrequency and look-ahead pruning to reduce the search space. Compared to Apriori, Max-miner can improve the performance of frequent pattern-mining by over two orders of magnitude. **H-mine** [152] overcomes the drawbacks of the FP-Growth algorithm (of requiring a large tree data structure) by generating a hyperlinked data structure, called H-struct. The algorithm scans the database once to create the H-struct, and from then onwards only needs to access this structure. By introducing pruning strategies, H-mine discovers only maximal frequent itemsets which makes it more space efficient than the FP-Growth.

Even though all four algorithms satisfy the functional requirements of our attack method, according to the preliminary experiments we conducted both max-miner and h-mine outperformed other two algorithms in terms of runtime and memory consumption. Max-miner, on the other hand, performed slightly better compared to h-mine in terms of memory consumption. Therefore, we used max-miner in the experimental evaluation of our attack.

7.4 Pattern-mining based Frequent Q-gram Identification

In the first step of our attack we identify the set of co-occurring bit positions in the BF database $\mathbf{B}$ that correspond to the frequent q-grams that occur in the plaintext database $\mathbf{V}$ [43]. We then refine these identified sets of bit positions for frequent q-grams by finding q-grams with potentially wrong bit positions assigned to them, as we describe in Section 7.4.2.

7.4.1 Identifying Co-occurring Bit Positions

As we describe in this section, Algorithm 7.1 [43] employs pattern-mining to find maximal frequent patterns of bit positions in $\mathbf{B}$ that correspond to frequent q-grams in $\mathbf{V}$. In line 1 the algorithm first converts $\mathbf{B}$ from its row storage (one BF per record) into a column-wise format $\mathbf{B}_c$ of l bit arrays each of length $|\mathbf{B}|$ to allow efficient access to individual bit positions. In line 3 the plaintext values $v \in \mathbf{V}$ are converted into a list of q-gram sets $\mathbf{q} \in \mathbf{Q}$ per record, where $\mathbf{Q}$ is the list of q-gram sets of all records in $\mathbf{V}$. In line 4 we initialise the list $\mathbf{F}$ of identified frequent q-grams and their sets of co-occurring BF bit positions, and the two lists $\mathbf{A}^+$ and $\mathbf{A}^-$ of must and cannot have q-gram sets assigned to BFs in $\mathbf{B}$, respectively. We also initialise the two sets $\mathbf{m}$ and $\mathbf{n}$ of must and cannot include q-grams (described below), and the list $\mathbf{k}$ which will hold the estimated numbers of hash functions, k_{est}.

In line 5 the row partition filter, $\mathbf{r}$, is initialised so all BFs are considered in the first iteration (i.e. the first partition contains all BFs in $\mathbf{B}$). Similarly, the column filter $\mathbf{c}$ is initialised so all bit positions of a BF are considered in the first partition. In line 6 we then initialise the queue Q with the first partition that contains all BFs and all bit positions.

The main loop of the algorithm starts in line 7 where in each iteration the largest partition in Q will be processed based on the partition size ps_i. A tuple is retrieved from the queue containing $\mathbf{c}_i$ (the column filter of which bit positions in $\mathbf{B}_c$ to consider in this iteration), $\mathbf{r}_i$ (the row filter of which BFs in $\mathbf{B}_c$ to consider), and $\mathbf{m}_i$ and $\mathbf{n}_i$ (the sets of q-grams that a q-gram set in $\mathbf{Q}$ must and cannot contain). The function *GetTwoMostFreqQGrams*() in line 9 returns the two most frequent q-grams, q_1 and q_2, and their respective frequencies, f_1 and f_2 (with $f_1 \geq f_2$), conditioned to contain all q-grams in $\mathbf{m}_i$ and no q-gram from $\mathbf{n}_i$. The percentage difference, d_i, between the frequencies f_1 and f_2 is then calculated (line 10). Only if d_i is at least the minimum percentage threshold d will partition i be processed. This ensures that q_1 occurs more frequently than q_2 in the considered partition as a potential frequent q-gram.

In line 12, the average frequency of q_1 and q_2 is converted into a minimum support count s_i of required 1-bits in the bit arrays $\mathbf{B}_c$ to be used for frequent pattern-mining. As detailed in Proposition 7.3.1, this is because any set of co-occurring bit positions in $\mathbf{B}_c$ that potentially can encode q_1 must have 1-bits in at least s_i common BFs (rows) in $\mathbf{B}_c$. The function *GetMaxFreqCoOccurBitPos*() in line 13 employs maximal frequent pattern-mining, as described in Section 7.3.3, on the bit positions in $\mathbf{B}_c$, where bit positions correspond to items and sets of co-occurring bit positions to itemsets. The aim of frequent pattern-mining is to find the maximal set (likely of size k) of co-occurring bit positions, $\mathbf{f}_i$, in $\mathbf{B}_c$ that have 1-bits in at least s_i common BFs. A set of two or more bit positions are co-occurring frequently if the bit-wise AND operation applied to their corresponding bit vectors in $\mathbf{B}_c$ has a Hamming weight of at least the minimum support s_i. As discussed in Section 7.3.3 we employ the max-miner algorithm for identifying maximal frequent bit patterns efficiently.

The function *GetMaxFreqCoOccurBitPos*() uses the parameters $\mathbf{c}_i$ and $\mathbf{r}_i$ to select a subset of BF bit positions (columns) and BFs from $\mathbf{B}_c$ (rows) to be considered in partition i. If a non-empty set of co-occurring bit positions $\mathbf{f}_i$ is returned by the function, then it is added together with the identified most frequent q-gram q_1 of this partition to the list of frequent q-grams $\mathbf{F}$ (line 15). This is because we have learned that the bit positions in $\mathbf{f}_i$ must encode q_1.

In line 16, we add the size of $\mathbf{f}_i$ (the number of co-occurring bit positions) to the list $\mathbf{k}$ of estimated number of hash functions, assuming that the pattern-mining algorithm was able to identify all positions that encode the frequent q-gram q_1. In line 17 we then remove the identified bit positions in $\mathbf{f}_i$ from the set $\mathbf{c}_i$ of positions to be considered in the following iterations. Such removal of bit positions will lead to higher precision of the identified bit positions (compared to when all bit positions are considered) because it is highly likely that only one frequent q-gram is assigned to each of these bit positions.

In the function $UpdateRowFilter()$, in line 18, based on the current set $\mathbf{r}_i$ of BFs that have been considered in this iteration, we generate the two new subsets $\mathbf{r}_i^+$ of BFs that do contain q_1 (have 1-bits in all positions in $\mathbf{f}_i$) and $\mathbf{r}_i^-$ of BFs that cannot contain q_1 (have at least one 0-bit in a position in $\mathbf{f}_i$), where $\mathbf{r}_i^- = \mathbf{r}_i \setminus \mathbf{r}_i^+$. We then add the frequent q-gram q_1 to the sets of must have (line 19) and cannot have (line 20) q-gram sets, $\mathbf{A}^+$ and $\mathbf{A}^-$, for all BFs in $\mathbf{r}_i^+$ and $\mathbf{r}_i^-$.

In lines 21 to 24 we generate two new partitions (if their size is at least the minimum partition size m), where in line 22 the new partition consists of those BFs that contain the frequent q-gram q_1 (so q_1 is added to the must have q-gram set $\mathbf{m}_i$), and in line 24 the new partition consists of those BFs that do not contain q_1 (and therefore q_1 is added to the set $\mathbf{n}_i$ of cannot have q-grams). In line 25, we then sort the queue Q such that the largest partition is again at the beginning of the queue.

In line 26 we estimate the number of hash functions, k_{est}, as the most common value of the sizes of the frequent bit position sets $\mathbf{f}_i$ identified in all iterations (as collected in the list $\mathbf{k}$). This is additional knowledge about the encoding parameters we gain from the pattern-mining process, and we use k_{est} in Algorithm 7.5 (as detailed in Section 7.6) to improve the accuracy of the attack. Finally, Algorithm 7.1 returns the list of frequent q-grams $\mathbf{F}$ and their identified sets of BF bit positions, and the two lists $\mathbf{A}^+$ and $\mathbf{A}^-$ that contain for each BF in $\mathbf{B}$ a set of frequent q-grams that must have (in $\mathbf{A}^+$) or that cannot have (in $\mathbf{A}^-$) been hashed to that BF. Both $\mathbf{A}^+$ and $\mathbf{A}^-$ are used in the second step, and $\mathbf{F}$ is used in second and third steps of our attack method, to identify additional q-grams, as we describe in the following sections.

7.4.2 Finding Q-grams with Inaccurate Bit Positions

In this section we propose a refinement step to identify q-grams in $\mathbf{F}$ with wrong bit positions assigned to them. Algorithm 7.1 returns a set of frequent q-grams and their corresponding bit positions, $\mathbf{F}$. Although these identified bit positions have high accuracy as we illustrate in Section 7.9, because of possible collisions in bit positions when encoding multiple q-grams into a single BF, some of the identified bit positions for certain q-grams can be wrong. When using these q-grams and bit positions in the later steps of our attack to expand the set of identified q-grams and to reidentify plaintext values (as detailed in Sections 7.5 to 7.7), having inaccurate bit positions will likely lead to inaccurate reidentification results. Therefore it is important to identify those q-grams in $\mathbf{F}$ with wrong bit positions assigned to them.

Lemma 7.4.1. *Following Proposition 7.3.1, assuming a q-gram $q \in \mathbf{F}$ occurs in $n_q = |\mathbf{V}_q|$ records in plaintext database $\mathbf{V}$, with $|\mathbf{V}| = n$ and $n_q < n$, then there must be a set of BFs $\mathbf{B}_q = \{b \in \mathbf{B} : \forall p \in \mathbf{F}[q] : b[p] = 1\}$ with $|\mathbf{B}_q| = m_q$ and $|\mathbf{B}| = m$, such that $|\frac{n_q}{n} - \frac{m_q}{m}| < \theta$, where θ is a maximum percentage difference between the sizes of the two subsets $\mathbf{V}_q$ and $\mathbf{B}_q$.*

According to the Lemma 7.4.1, we assume that for an identified frequent q-gram $q \in \mathbf{F}$, the set of plaintext values which contain q and the set of BFs which encode q should be similar in size with an acceptable size difference limit θ. If there is a

considerably large difference between the sizes of those two sets, then we assume that one or more bit position(s) were wrongly assigned to q in $\mathbf{F}$. Following this condition, by analysing each q-gram $q \in \mathbf{F}$ individually using both $\mathbf{V}$ and $\mathbf{B}$ we can identify which q-grams result in considerable size differences between BF subsets and their corresponding plaintext subsets due to wrongly identified bit positions. For identifying exactly which bit position(s) is/are wrong we use the following:

Lemma 7.4.2. *For two given identified q-grams, $q_1, q_2 \in \mathbf{F}$, where $\mathbf{B}_{q_2} = \{\mathbf{b} \in \mathbf{B} : \forall p \in \mathbf{F}[q_2] : \mathbf{b}[p] = 1\}$, and assuming q_1 does contain one or more incorrect bit position(s) and q_2 does not, with conditional probability $\rho(q_1|q_2) \geq \alpha$, where α is a defined lower conditional probability limit between two q-grams, for all correct bit positions in q_1, the following condition must hold: $\forall p \in \mathbf{F}[q_1], hamming(\mathbf{B}_{q_2}[p]) \geq \rho(q_1|q_2) \cdot |\mathbf{B}_{q_2}| \cdot \beta$, where $hamming()$ is the Hamming weight of a given bit position and $0 < \beta \leq 1$ is a defined confidence threshold value.*

By following Lemma 7.4.1 we are able to identify q-grams with correct and wrong bit positions. Then, according to Lemma 7.4.2, we first pair q-grams that contain wrong bit positions (q_1) with q-grams that contain correct bit positions (q_2), where we only select q-gram pairs that have a conditional probability $\rho(q_1|q_2)$ over a defined limit α. We next obtain the BF set, $\mathbf{B}_{q_2}$, which encodes the correct q-gram q_2 and analyse each bit position assigned to q_1 using $\mathbf{B}_{q_2}$. According to the conditional probability $\rho(q_1|q_2)$ obtained using $\mathbf{V}$, if a bit position assigned to q_1 is correct, that bit position in $\mathbf{B}_{q_2}$ should have at least a Hamming weight of $\rho(q_1|q_2) \cdot |\mathbf{B}_{q_2}| \cdot \beta$. If a bit position does not satisfy this condition then we label that bit position as a wrong position and remove it from the set of assigned bit positions for $q_1 \in \mathbf{F}$.

Using the above two conditions the attack is able to find which identified q-grams have incorrect bit positions assigned to them, and which of those bit positions are wrong. An adversary can use these conditions to remove such wrong bit positions or even entire q-grams that were wrongly identified to improve the accuracy of the q-gram sets in $\mathbf{F}$. Note that this step can also be used at the end of both the second and third steps (Sections 7.5 and 7.6) to analyse the sets of q-grams and their corresponding bit positions identified in these steps. The accuracy of the identified q-grams with wrong bit positions are shown and discussed in Section 7.9.

7.5 Language Model based Q-gram Set Expansion

In the second step of our attack we expand the list of frequent q-grams, $\mathbf{F}$, using a probabilistic language model by identifying non-frequent q-grams in $\mathbf{Q}$ (and their corresponding BF bit positions) that have a high conditional probability of co-occurring with one of the frequent q-grams in $\mathbf{F}$. In this section we first discuss how we generate the language model and then how we use this model combined with pattern-mining to identify additional non-frequent q-grams and the sets of corresponding BF bit positions they are hashed into. We also discuss and justify the results obtained by this model via two probability calculations.

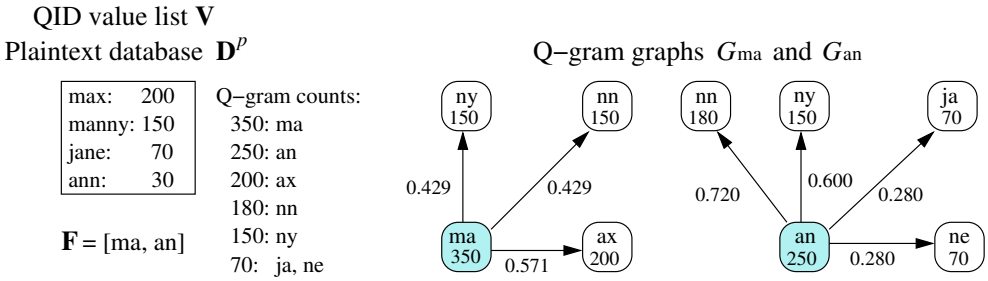


Figure 7.3: Example q-gram graphs generated from four plaintext values containing six q-grams, where 'ma' and 'an' are frequent q-grams identified in the first step of our attack. Edges between q-grams show conditional probabilities of the destination q-gram assuming the source q-gram occurs in a value in **V**. For each q-gram in **F**, in the second step of our attack we select the non-frequent q-grams with the highest conditional probabilities, such as 'ax' for 'ma' and 'nn' for 'an'. As described in Section 7.5, we then apply pattern-mining on subsets of BFs in **B** where we know they contain a frequent q-gram in **F** to identify the bit positions that can encode a selected non-frequent q-gram.

7.5.1 Generating the Language Model Graph

As illustrated in Figure 7.3, conceptually we build a star graph G for each frequent q-gram $q_f \in \mathbf{F}$, where the centre node is the frequent q-gram $q_f \in \mathbf{F}$ as identified in the first step of our attack, and leaf nodes are other q-grams q_i that co-occur with q_f in at least one value $v_j \in \mathbf{V}$. The node attributes are the frequencies of q-grams q_i, calculated as $f_i = |\{q_i \in \mathbf{q}_j : \mathbf{q}_j \in \mathbf{Q}\}|$, where $\mathbf{q}_j$ is the q-gram set for plaintext value v_j as generated in line 3 in Algorithm 7.1.

A directed edge, $q_f \rightarrow q_i$ in G represents the conditional probability $\rho(q_f|q_i)$ of q_i occurring in a value $v_j \in \mathbf{V}$ given the frequent q-gram q_f occurs in v_j, calculated as $\rho(q_f|q_i) = f_i/f_f$. For example, in G_{ma} in Figure 7.3, the conditional probability of 'ax' occurs given 'ma' occurs is $200/350 = 0.571$.

Algorithm 7.2 details the steps for generating the language model **L**. The algorithm takes as input the set of frequent q-grams **F** and the list of q-gram sets, **Q**, as generated in Algorithm 7.1. For each frequent q-gram $q_f \in \mathbf{F}$, in line 3 we create a new graph G which only contains q_f as a node but no edges. The algorithm then loops over all q-gram sets $\mathbf{q}_j \in \mathbf{Q}$ (in line 4), and if a q-gram set does contain q_f its count, denoted as f_f, is increased by 1 in line 6. For all other q-grams $q_i \in \mathbf{q}_j$, we first add q_i as a new node with count 0 if it is not in the graph (lines 8 to 10), and then increase its count by 1 in line 11.

In lines 12 to 14 of the algorithm we then add the edges from q_f to all other q-grams q_i in the graph (line 13) where an edge attribute is the conditional probability calculated in line 14 as the count for a q_i divided by the count of q_f. Finally we add q_f and the graph G to the language model **L** in line 15, and return **L** in line 16.

Once **L** is generated, we can use it to expand the set of identified q-grams. The idea of this expansion step is that for each frequent q-gram, q_f, we can use non-

Algorithm 7.2: Generate language model for q-gram set expansion

Input : **F**: List of identified q-grams and their co-occurring BF bit positions
Q: List of q-gram sets, one per plaintext value in **V**

Output: **L**: Language model as one graph G per frequent q-gram $q_f \in \mathbf{F}$

1 $\mathbf{L} \leftarrow [\,]$ // Initialise the language model graph list
2 **for** $q_f \in \mathbf{F}$ **do**: // Main loop over all identified frequent q-grams
3 $V \leftarrow \{q_f\}$; $E \leftarrow \emptyset$; $G \leftarrow (V, E)$ // Initialise a graph with the centre node
4 **for** $\mathbf{q}_j \in \mathbf{Q}$ **do**: // Loop over each q-gram set of plaintext values in **V**
5 **if** $q_f \in \mathbf{q}_j$ **then**: // Check if the frequent q-gram is in the current q-gram set
6 $G.V[q_f] \leftarrow G.V[q_f] + 1$ // Increment centre node (q_f) frequency by 1
7 **for** $q_i \in \mathbf{q}_j \setminus \{q_f\}$ **do**: // Loop over the non-frequent q-grams
8 **if** $q_i \notin G.V$ **then**: // If q_i is not a node in G
9 $G.V \leftarrow G.V \cup \{q_i\}$ // Add non-frequent q-gram to graph nodes
10 $G.V[q_i] \leftarrow 0$ // Initialise node (q_i) frequency
11 $G.V[q_i] \leftarrow G.V[q_i] + 1$ // Increment node (q_i) frequency by 1
12 **for** $q_i \in G.V \setminus \{q_f\}$ **do**: // Loop over each non-frequent node in the graph
13 $G.E \leftarrow G.E \cup \{(q_f, q_i)\}$ // Add edge between q_f and q_i to G
14 $G.E[q_f, q_i] \leftarrow G.V[q_i]$ / $G.V[q_f]$ // Calculate the conditional probability
15 $\mathbf{L}.append([q_f, G])$ // Add q-gram graph to the language model
16 **return** **L**

frequent q-grams in G that have high conditional probabilities of co-occurring with q_f to identify the bit positions that these non-frequent q-grams are hashed into. In the following section we describe this approach in detail.

7.5.2 Identifying Further Q-grams using the Language Model

In Algorithm 7.3 we outline the steps in the q-gram sets expansion phase based on the language model **L** generated in Algorithm 7.2. We first initialise the expanded sets $\mathbf{F}_E$, $\mathbf{A}_E^+$, and $\mathbf{A}_E^-$ of identified q-grams, and must and cannot have q-gram sets as those obtained in the first step of our attack, as well as an empty queue Q. In line 3, a column filter, **c**, is created to contain only those BF bit positions that have no frequent q-gram from **F** assigned to them.

We then populate Q in lines 4 to 7 with tuples that contain for each frequent q-gram $q_f \in \mathbf{L}$ (1) the frequency f_f of q_f as the number of q-gram sets in **Q** that contain q_f; (2) the q-gram q_f; (3) its graph G as generated in Algorithm 7.2; and (4) the row filter $\mathbf{r}_f$ (in line 4) which contains the indices b of all BFs in **B** that we believe contain q_f, i.e. the BF subset $\mathbf{B}_f$, where these BFs have a 1-bit in all positions in $\mathbf{f}_f$ from **F**. In line 8, we sort the queue Q with the highest q-gram frequency f_f occurring first.

The main loop of Algorithm 7.3 starts in line 9, and in line 10 the first tuple in Q is retrieved. In line 11, the graph nodes, $G.V$, are sorted according to the conditional probabilities assigned to the edges such that the non-frequent q-gram, q_i, with the highest conditional probability, $\rho(q_f|q_i)$, is considered first in the following loop. In line 12 we iterate over pairs of q-grams and their conditional probabilities, $(q_i, \rho(q_f|q_i))$, where in each iteration we retrieve two consecutive q-grams from G.

Algorithm 7.3: Identify further q-grams using the language model

Input : $\mathbf{B}$: List of BFs from the sensitive encoded database
$\mathbf{B}_c$: Column-wise storage of BF database $\mathbf{B}$
$\mathbf{F}$: List of identified frequent q-grams and their co-occurring BF bit positions
$\mathbf{A}^+$: List of must have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{A}^-$: List of cannot have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{Q}$: List of q-gram sets, one per plaintext value in $\mathbf{V}$
$\mathbf{L}$: Language model as one graph G per frequent q-gram $q_f \in \mathbf{F}$
k_{est}: Estimated number of hash functions used to encode q-grams
d: Minimum percentage difference between two q-grams

Output: $\mathbf{F}_E$: Expanded list of identified q-grams and their BF bit positions
$\mathbf{A}_E^+$: Expanded list of must have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{A}_E^-$: Expanded list of cannot have q-gram sets assigned to BFs in $\mathbf{B}$

1 $\mathbf{F}_E \leftarrow \mathbf{F}$, $\mathbf{A}_E^+ \leftarrow \mathbf{A}^+$, $\mathbf{A}_E^- \leftarrow \mathbf{A}^-$ // Initialise new lists with previous lists
2 $Q \leftarrow [\,]$ // Initialise a queue
3 $\mathbf{c} \leftarrow \{p : 1 \leq p \leq |\mathbf{B}_c|\} \setminus \{\cup p : \forall p \in \mathbf{f}_i : \forall \mathbf{f}_i \in \mathbf{F}\}$ // Initialise the column filter
4 **for** $(q_f, G) \in \mathbf{L}$ **do**: // Loop over each frequent q-gram graph in $\mathbf{L}$
5 $\mathbf{r}_f \leftarrow \{b : 1 \leq b \leq |\mathbf{B}| : \forall p \in \mathbf{f}_f : \mathbf{B}[b][p] \leftarrow 1 : \mathbf{f}_f \in \mathbf{F}\}$ // Initialise the row filter
6 $f_f \leftarrow G[q_f]$ // Get the frequency of the frequent q-gram
7 $Q.append((f_f, q_f, G, \mathbf{r}_f))$ // Add the obtained values to Q
8 $Q.sort_most_freq_first()$ // Sort Q to get the most frequent q-gram first
9 **while** $|Q| > 0$ **do**: // Main loop: run until the queue is empty
10 $(f_f, q_f, G, \mathbf{r}_f) \leftarrow Q.pop()$ // Get the first item (q-gram graph) in the queue
11 $G.V.sort_high_cond_prob()$ // Sort nodes in G in descending order of cond. prob.
12 **for** $[(q_i, \rho(q_f|q_i)), (q_{i+1}, \rho(q_f|q_{i+1}))] \in G.V \setminus \{q_f\}$ **do**: // Loop over each q-gram pair
13 $d_i \leftarrow 2(\rho(q_f|q_i) - \rho(q_f|q_{i+1}))/(\rho(q_f|q_i) + \rho(q_f|q_{i+1})) \cdot 100$ // Get the percentage diff.
14 **if** $d_i \geq d$ **then**: // Only continue if percentage difference is large enough
15 $s_i \leftarrow |\mathbf{r}_f| \cdot (p_{f,i} + p_{f,i+1})/2$ // Minimum support for frequent pattern-mining
16 $\mathbf{f}_i \leftarrow GetMaxFreqCoOccurBitPos(\mathbf{B}_c, s_i, \mathbf{c}, \mathbf{r}_f)$ // Run pattern-mining
17 **if** $0 < |\mathbf{f}_i| \leq k_{est}$ **then**: // A set of frequent bit positions is identified
18 $\mathbf{F}_E.append((q_i, \mathbf{f}_i))$ // Assign bit positions to q_i
19 $\mathbf{c} \leftarrow \mathbf{c} \setminus \mathbf{f}_i$ // Remove identified bit positions
20 $\mathbf{r}_i^+, \mathbf{r}_i^- \leftarrow UpdateRowFilter(\mathbf{f}_i, \mathbf{r}_f)$ // Based on bit positions in f_i
21 $\forall b \in \mathbf{r}_i^+ : \mathbf{A}_E^+[b] \leftarrow \mathbf{A}_E^+[b] \cup \{q_i\}$ // Assign q_i to must have q-gram sets
22 $\forall b \in \mathbf{r}_i^- : \mathbf{A}_E^-[b] \leftarrow \mathbf{A}_E^-[b] \cup \{q_i\}$ // Assign q_i to cannot have q-gram sets
23 **else**: // // If $d_j < d$
24 **break** // Terminate the loop
25 **return** $\mathbf{F}_E$, $\mathbf{A}_E^+$, $\mathbf{A}_E^-$ // Return expanded lists

Similar to Algorithm 7.1, in line 13 we calculate the percentage difference d_i between the two probabilities, $\rho(q_f|q_i)$ and $\rho(q_f|q_{i+1})$, and only if this difference is at least the minimum difference d are we considering the q-gram q_i in the following steps. Otherwise we end the loop in line 24. This is because in such a case the non-frequent q-grams for q_f do not have distinct enough frequencies to allow precise identification of their corresponding bit positions. In line 15, we calculate the minimum required support s_i for q_i based on the average of $\rho(q_f|q_i)$ and $\rho(q_f|q_{i+1})$ and the size of the BF subset $\mathbf{B}_f$ for q_f using the BF row filter $\mathbf{r}_f$. In line 16 we then call the pattern-mining algorithm on this subset $\mathbf{B}_f$. If a set of frequently co-occurring bit positions $\mathbf{f}_i$ is identified, and not more than k_{est} bit positions co-occur, then we assign q-gram

q_i to this new set $\mathbf{f}_i$ of bit positions and add the pair $(q_i, \mathbf{f}_i)$ to the expanded list of identified q-grams, $\mathbf{F}_E$, in line 18.

Similar to Algorithm 7.1, we then remove the newly identified bit positions, $\mathbf{f}_i$, from the column filter set $\mathbf{c}$ in line 19, and update the row filter sets $\mathbf{r}_i^+$ and $\mathbf{r}_i^-$ of BFs that must and cannot contain q_i in line 20. We also add the identified q-gram q_i to the must have and cannot have q-gram sets in the lists $\mathbf{A}_E^+$ and $\mathbf{A}_E^-$ for all BFs in the sets $\mathbf{r}_i^+$ and $\mathbf{r}_i^-$ (lines 21 and 22). We then loop back to line 12 to process the next pair of q-grams (q_i, q_{i+1}) of the frequent q-gram q_f. The algorithm terminates once all frequent q-grams in $\mathbf{L}$ have been processed and the queue Q becomes empty.

7.5.3 Probability of False Positives for Bloom Filter Subsets

In order to identify additional q-grams with high accuracy using the language model we need to make sure that all BFs in the subset $\mathbf{B}_f$ actually do encode the frequent q-gram q_f. If $\mathbf{B}_f$ is not selected accurately, then our q-gram expansion step will lead to wrongly identified additional q-grams which will then result in a reduced reidentification accuracy of plaintext values.

For a given BF in $\mathbf{B}_f$ that has a 1-bit in all k bit positions for q_f, there is a possibility that these k 1-bits have been set by the other non-frequent q-grams that are hashed into the BF that we believe encodes q_f. In order for this to happen, assuming e other non-frequent q-grams were hashed (k times each) into the same BF, then each of the k bit positions of q_f needs to collide at least once with one of the $e \cdot k$ hash mappings of these other e q-grams. We therefore need to calculate the probability of $m = e \cdot k$ hash mappings to collide with all k bit positions of q_f given a BF of length l bits. This corresponds to the probability of having l urns, where k of these urns contain a blue ball (i.e. q_f is hashed into) and where $e \cdot k$ red balls are thrown into these urns with equal probabilities such that at the end each of the urns that contains a blue ball will also contain at least one red ball (i.e. have one of the e q-grams $q_i \neq q_f$ hashed into).

Similar to Equation 7.1, we denote with $\mathbf{z} = [z_1, \ldots, z_m]$ a set of indicator variables, where $m = e \cdot k$ is the number of hash mappings (of $e \geq 1$ other q-grams hashed k times each). A value $z_i = 1$ means the current hash mapping is assumed to collide with a bit position where the frequent q-gram q_f has previously been hashed into, and where this bit has so far not been hit by a previous hash mapping of any of the non-frequent q-grams. A value $z_i = 0$ means the current hash mapping does not hit with any of the k bit positions where q_f has been hashed into (that so far have not been hit by a previous hash mapping of any of the non-frequent q-grams).

Given there are m hash mappings of e non-frequent q-grams, there are $\binom{m}{k}$ possibilities (combinations) that k of the m hash mappings collide with the k bit positions of the q-gram q_f. For each of these $\binom{m}{k}$ combinations of k hash collisions we calculate the probability for a given sequence $\mathbf{z}_j$ (assuming $\sum_{i=1}^{m} z_i = k$, i.e. k of the $z_i = 1$) as:

$$\rho(\mathbf{z}_j) = \prod_{x=1}^{m} \left(\frac{l - \left(k - \sum_{y=1}^{x-1} z_y\right)}{l} \right)^{(1-z_x)} \cdot \left(\frac{k - \sum_{y=1}^{x-1} z_y}{l} \right)^{z_x} . \tag{7.2}$$

The first factor in the product corresponds to the case where a hash mapping does not collide with one of the k bit positions where the frequent q-gram q_f was hashed into (i.e. no collision), or the mapping collides with a bit position that was already hit by one of the previous hash mappings (of any of the not frequent q-grams). The second factor corresponds to the case where a hash mapping hits a bit position where the frequent q-gram q_f was hashed into and where this bit position has not been hit by any of the previous hash mappings. For a certain hash mapping x, with $1 \leq x \leq m$, the calculated fractions depend upon the number of previous collisions, which are calculated in $\sum_{y=1}^{x-1} z_y$. When $x = 1$ we set the sums inside both factors to 0 since $x = 1$ indicates the first hash mapping, and no other hash mappings were performed prior to that. In case the second factor becomes $0/l$ (where the sum over the z_y becomes k) we set $0^0 = 1$.

To calculate the overall probability $\rho(l, k, e)$ that e other q-grams have generated the k 1-bit positions identified for the frequent q-gram q_f in a BF of length l, we need to sum the probability $\rho(\mathbf{z}_j)$ from Equation 7.2 over all $e \cdot \binom{k}{k}$ corresponding combinations of indicator variable sequences $\mathbf{z}_j$:

$$\rho(l, k, e) = \sum_{j=1}^{\binom{e \cdot k}{k}} \rho(\mathbf{z}_j), \text{ subject to } \sum_{i=1}^{e \cdot k} z_i = k, \tag{7.3}$$

where $z_i \in \mathbf{z}_j$. For commonly used values of k [61, 177], such as $10 \leq k \leq 50$, the number of such combinations makes the calculation of this overall probability computationally highly expensive. We therefore calculate an approximation which will always provide an overall probability equal to or larger than the exact value as:

$$\rho(l, k, e)_{est} = \binom{e \cdot k}{k} \cdot \prod_{i=1}^{k} \left(\frac{k - i - 1}{l} \right). \tag{7.4}$$

In this approximation, all non-collisions in Equation 7.2 (first factor) are replaced with a value of 1 (not shown in Equation 7.4) because the non-collision fractions in Equation 7.2 range from $(l - k)/l$ to l/l, where l is usually much larger than k. As a result the product of the non-collisions in Equation 7.2 (first factor) will always be lower than 1 while in the approximation it is assumed to be 1. Since $\sum_{i=1}^{m} z_i = k$, i.e. the event $z_i = 1$ happens only k times, the sum inside the second factor in Equation 7.2 will only go from 1 to k. Furthermore, the event $z_i = 0$ happens $e - k$ times and in such an event the second factor in Equation 7.2 will be equal to 1. Therefore the second factor in Equation 7.2 can be simplified as follows:

$$\prod_{x=1}^{e \cdot k} \left(\frac{k - \sum_{y=1}^{x-1} z_y}{l} \right)^{z_x} = \prod_{i=1}^{k} \left(\frac{k - i - 1}{l} \right). \tag{7.5}$$

Hence the second factor in this approximation is the same product of fractions as in Equation 7.2.

Table 7.3: Probabilities calculated using Equation 7.4 of hashing a q-gram into the same k 1-bit positions that were previously hit by at least one hash mapping of e other q-grams, for different values of k and e, and with $l = 1{,}000$.

	$e = 1$	$e = 2$	$e = 3$	$e = 4$	$e = 8$	$e = 9$	$e = 10$
$k = 10$	0.0	3.62*e*–24	6.70*e*–19	1.09*e*–16	1.44*e*–12	5.97*e*–12	2.07*e*–11
$k = 20$	0.0	2.43*e*–42	3.35*e*–31	1.01*e*–26	2.01*e*–18	3.50*e*–17	4.26*e*–16
$k = 30$	0.0	2.65*e*–58	3.13*e*–41	1.78*e*–34	5.27*e*–22	3.84*e*–20	1.64*e*–18
$k = 50$	0.0	3.04*e*–86	3.06*e*–57	6.12*e*–46	4.03*e*–25	5.18*e*–22	2.70*e*–19

In Table 7.3 we show calculated collision estimates based on Equation 7.4 for different numbers of hash functions, k, and other q-grams, e, that have been hashed into the same BF. As can be seen, all these probabilities are very low, which confirms that the subsets $\mathbf{B}_f$ of BFs we select for a given frequent q-gram q_f (based on the bit positions in $\mathbf{f}$ for q_f) will be highly accurate.

7.5.4 Probability of Collisions between Non-Frequent and Frequent Bit Positions

One important aspect about Algorithm 7.3 is that we only consider those BF bit positions that have not yet been identified to encode a frequent q-gram in $\mathbf{F}$ (in line 3 where we generate the column filter set $\mathbf{c}$). Hence, there is a high possibility of missing some bit positions that encode a non-frequent q-gram q_i if these positions have previously been identified to encode any of the frequent q-grams in $\mathbf{F}$. As a result, the sizes of the sets $\mathbf{f}_i$ (in line 16 of Algorithm 7.3) returned by the pattern-mining algorithm are likely to be less than k_{est}. This will lead to a reduced recall of identified bit positions in the q-gram set expansion step, as our experimental results in Section 7.9 confirm.

We can calculate the probability that a q-gram q_i shares u bit positions, with $0 \leq u \leq k$, with any $q_f \in \mathbf{F}$ as:

$$\rho(l, k, n_F, x) = \binom{k}{u} \cdot \left(\frac{k \cdot n_F}{l}\right)^{u} \cdot \left(\frac{l - k \cdot n_F}{l}\right)^{(k-u)}, \tag{7.6}$$

where $n_F = |\mathbf{F}|$. The first factor in the product corresponds to the number of possibilities (combinations) that u positions of q_i can be drawn from k positions. The second factor is the probability that the selected u positions of q_i are hashed into any of the already identified bit positions from frequent q-grams in $\mathbf{F}$. The last factor of the product is the probability that other positions of q_i are hashed into any of the bit positions that have not been identified as positions that encode any frequent q-gram in $\mathbf{F}$. Note that in Equation 7.6 we are calculating the probabilities for the worst-case scenario where each frequent q-gram has its own unique bit positions. Hence this probability calculation indicates the upper bound for the probability of collisions.

Table 7.4: Upper bound probabilities of other q-grams sharing u 1-bit positions as set previously by n_F frequent q-grams in **F** (as calculated in Equation 7.6) for different k where $l = 1{,}000$ bits. Worst results (highest probabilities) per row are shown in bold font.

n_F	k	$u = 0$	$u = 1$	$u = 2$	$u = 4$	$u = 6$	$u = 7$	$u = 8$
5	10	**0.599**	0.315	0.075	0.001	2.67*e*–6	8.04*e*–8	1.59*e*–9
	15	0.311	**0.378**	0.214	0.018	4.42*e*–4	4.6*e*–5	3.73*e*–6
	30	0.008	0.040	0.103	**0.203**	0.137	0.083	0.042
	50	5.66*e*–7	9.44*e*–6	7.71*e*–5	0.002	0.012	0.026	**0.046**
10	10	0.349	**0.387**	0.194	0.011	1.37*e*–4	8.74*e*–6	3.64*e*–7
	15	0.087	0.231	**0.286**	0.115	0.013	0.003	5.28*e*–4
	30	2.25*e*–5	2.89*e*–4	0.002	0.021	0.083	0.122	**0.150**
	50	8.88*e*–16	4.44*e*–14	1.08*e*–12	2.04*e*–10	1.41*e*–8	8.87*e*–8	**4.77*e*–7**

In Table 7.4 we show example collision probabilities for different values of k and n_F. The results in this table show a lower recall can be expected for the q-grams identified in the expansion step, especially when k and n_F increase.

7.6 Partitioning based Q-gram Set Expansion

Due to the limitation of getting a lower recall when using the language model (as shown theoretically in Section 7.5.4 and empirically in Section 7.9), we next describe an approach based on partitioning the BF and plaintext databases into subsets to obtain bit positions of q-grams with high precision and recall results. We first present the partitioning of both the plaintext and BF databases into multiple distinct partition pairs using different combinations of frequent q-grams from **F**. Then we describe how we can employ pattern-mining to analyse these pairs of partitions to further identify additional non-frequent q-grams and their corresponding bit positions.

7.6.1 Plaintext and Bloom Filter Partitioning

In Algorithm 7.4 we show the partitioning process of plaintext QID values and BFs based on frequent q-gram pairs. The inputs to the algorithm are **F**, the list of identified q-grams and their corresponding bit positions; **Q**, the list of q-gram sets; **B**, the list of BFs from the encoded database; and **n**, a list of length values which is used to create frequent q-gram combinations. The algorithm returns Q, a queue containing the generated BF and plaintext partition pairs along with their corresponding column filters. In line 1 we initialise Q and in lines 2 and 3 we loop over each combination length $n \in \mathbf{n}$ and get all possible frequent q-gram subsets of length n. For instance, a frequent q-gram set $\{sm, mi, th\} \in \mathbf{F}$ can be divided into a set of q-gram subsets of length $n = 2$ as $\mathbf{M} = \{\{sm, mi\}, \{sm, th\}, \{mi, th\}\}$.

Algorithm 7.4: Bloom filter and plaintext partitioning

Input : **F**: List of identified frequent q-grams and their co-occurring BF bit positions
Q: List of q-gram sets, each per plaintext value in **V**
B: List of BFs from the sensitive encoded database, one BF per record
n: A list of lengths to create q-gram combinations

Output: Q: A queue of identified BF and q-gram (plaintext) partitions and column filters

1 $Q \leftarrow [\,]$ // Initialise a partitioning model based queue
2 **for** $n \in \mathbf{n}$ **do**: // Loop to generate partitions
3 $\mathbf{M} \leftarrow GetAllCombinations(\mathbf{F}, n)$ // Frequent q-gram combinations of length n
4 $\mathbf{S} \leftarrow GetAllPermutations(n)$ // Get all permutations of a length n bit vector
5 **for** $\mathbf{m} \in \mathbf{M}$ **do**: // Loop over all subsets of frequent q-grams
6 **if** $isValid(\mathbf{m}, \mathbf{Q})$ **then**: // Check the validity of the combination $\mathbf{m}$
7 **for** $\mathbf{s} \in \mathbf{S}$ **do**: // Loop over all permutations of {0,1}
8 $\mathbf{c}_s \leftarrow \{p : 1 \leq p \leq |\mathbf{B}_c|\}$ // Initialise column filter
9 $\mathbf{r}_s \leftarrow \{b : 1 \leq b \leq |\mathbf{B}|\}$ // Initialise BF partition/row filter
10 $\mathbf{q}_s \leftarrow \{\mathbf{q} \in \mathbf{Q}\}$ // Initialise q-gram (plaintext) partition
11 **for** $s_i \in \mathbf{s}, q_i \in \mathbf{m}$ **do**: // Loop over all q-grams and bit values
12 **if** $s_i = 0$ // The q-gram cannot be in the partition
13 $\mathbf{r}_s \leftarrow \mathbf{r}_s \setminus \{\cup b : 1 \leq b \leq |\mathbf{B}| : \forall p \in \mathbf{f}_i : \mathbf{B}[b][p] \leftarrow 1 : \mathbf{f}_i \in \mathbf{F}[q_i]\}$
14 $\mathbf{q}_s \leftarrow \mathbf{q}_s \setminus \{\cup \mathbf{q} : q_i \in \mathbf{q} : \forall \mathbf{q} \in \mathbf{Q}\}$ // Update q-gram set
15 **else:** // The q-gram must be in the partition
16 $\mathbf{r}_s \leftarrow \mathbf{r}_s \cap \{\cup b : 1 \leq b \leq |\mathbf{B}| : \forall p \in \mathbf{f}_i : \mathbf{B}[b][p] \leftarrow 1 : \mathbf{f}_i \in \mathbf{F}[q_i]\}$
17 $\mathbf{q}_s \leftarrow \mathbf{q}_s \cap \{\cup \mathbf{q} : q_i \in \mathbf{q} : \forall \mathbf{q} \in \mathbf{Q}\}$ // Update q-gram set
18 $\mathbf{c}_s \leftarrow \mathbf{c}_s \setminus \{p : \forall p \in \mathbf{f}_i : \mathbf{f}_i \leftarrow \mathbf{F}[q_i]\}$ // Update the column filter
19 $Q.append((\mathbf{m}, \mathbf{s}, \mathbf{q}_s, \mathbf{r}_s, \mathbf{c}_s))$ // Add the obtained partition to Q
20 **return** Q // Return the final queue

Once we have obtained frequent q-gram subsets, we consider all instances of each subset to split the BF and plaintext databases into partitions. For example assuming a q-gram subset $\{q_1, q_2\}$, there exists four possible instances:

$$\{q_1, q_2\} = \begin{cases} (1,1), & \{q_1, q_1\} \\ (1,0), & \{q_1, \overline{q_2}\} \\ (0,1), & \{\overline{q_1}, q_2\} \\ (0,0), & \{\overline{q_1}, \overline{q_2}\} \end{cases} \tag{7.7}$$

In the first instance, (1,1), both q-grams must be in the partitions, in the second instance, (1,0), q_1 must be in the partitions and q_2 cannot be in the partitions, in the third instance, (0,1), q_2 must be in the partitions and q_1 cannot be in the partitions, and in the last instance, (0,0), both q-grams cannot be in the partitions. Note that $\overline{q_1}$, and $\overline{q_2}$ mean that the corresponding q-gram cannot be in a partitions. In general if there are $n = |\mathbf{m}|$ q-grams in a certain q-gram subset, $\mathbf{m} \in \mathbf{M}$, there can be 2^n BF and plaintext subset partitions.

The function $GetAllPermutations()$ in line 4 generates all these 2^n permutations, **S**, of {0,1} bit vectors of length $n \in \mathbf{n}$ to represent all instances of q-gram subsets. Then, in line 5, we loop over all q-gram subsets, $\mathbf{m} \in \mathbf{M}$. The function $isValid()$ in line 6 is a pruning step where we check if the current q-gram subset is a valid subset to be processed.

Algorithm 7.5: Identifying further q-grams using BF and plaintext partitions

Input : $\mathbf{B}$: List of BFs from the sensitive encoded database, one BF per record
$\mathbf{B}_c$: Column-wise storage of BF database $\mathbf{B}$
$\mathbf{F}_E$: Expanded list of identified q-grams and their BF bit positions
$\mathbf{A}_E^+$: Expanded list of must have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{A}_E^-$: Expanded list of cannot have q-gram sets assigned to BFs in $\mathbf{B}$
k_{est}: Estimated number of hash functions used to encode q-grams
k_{err}: Error limit for estimated number of hash functions
d: Min percentage difference between two most frequent q-grams
Q: The queue of identified BF and plaintext partitions

Output: $\mathbf{F}_F$: Final list of identified q-grams and their BF bit positions
$\mathbf{A}_F^+$: Final list of must have q-gram sets assigned to BFs in $\mathbf{B}$
$\mathbf{A}_F^-$: Final list of cannot have q-gram sets assigned to BFs in $\mathbf{B}$

$\mathbf{F}_F \leftarrow \mathbf{F}_E$, $\mathbf{A}_F^+ \leftarrow \mathbf{A}_E^+$, $\mathbf{A}_F^- \leftarrow \mathbf{A}_E^-$ // Initialise new lists with previous lists
$\mathbf{F}_N \leftarrow [\,]$ // Initialise a new list for identified q-grams
while $|Q| > 0$ **do**: // Runs until the queue is empty
  $(\mathbf{m}, \mathbf{s}, \mathbf{q}_s, \mathbf{r}_s, \mathbf{c}_s) \leftarrow Q.pop()$ // Get the first partition in the queue
  $\mathbf{q}_f \leftarrow getOtherQgramFrequencies(\mathbf{q}_s)$ // Calculate q-gram frequencies
  $\mathbf{q}_f.sort_most_freq_first()$ // Sort in descending order by their frequencies
  for $[(q_i, f_i), (q_{i+1}, f_{i+1})] \in \mathbf{q}_f$ **do**: // Loop over each q-gram pair
    $d_i \leftarrow 2(f_i - f_{i+1})/(f_i + f_{i+1}) \cdot 100$ // Get the percentage difference
    if $d_i \geq d$ **then**: // Only continue if percentage difference is large enough
      $s_i \leftarrow |\mathbf{r}_s| \cdot (f_i + f_{i+1})/2$ // Minimum support for frequent pattern-mining
      $\mathbf{f}_i \leftarrow GetMaxFreqCoOccurBitPos(\mathbf{B}_c, s_i, \mathbf{c}_s, \mathbf{r}_s)$ // Run pattern-mining
      if $k_{est} - k_{err} \leq |\mathbf{f}_i| \leq k_{est} + k_{err}$ **then**: // If $|\mathbf{f}_i|$ is within the range for k
        $\mathbf{F}_N[q_i].append(\mathbf{f}_i)$ // Add identified q-gram and its bit positions to $\mathbf{F}_N$
        $\mathbf{c}_s \leftarrow \mathbf{c}_s \setminus \mathbf{f}_i$ // Remove identified bit positions
    else: // The frequency difference is not large enough
      break // Exit the loop
for $q_i \in \mathbf{F}_N$ **do**: // Loop over each identified other q-gram
  $\mathbf{F}_i \leftarrow \mathbf{F}_N[q_i]$ // Get the list of identified bit position tuples
  if $|\mathbf{F}_i| = 1$ **then**: // Check if the list contains only one bit position tuple
    $\mathbf{f}_i \leftarrow \mathbf{F}_i.pop()$ // Get the first item in the list
  else: // The list contains more than one bit position tuple
    $\mathbf{f}_i \leftarrow GetMostFreqBitPos(\mathbf{F}_i, k_{est})$ // Get k_{est} most frequent bit positions
  $\mathbf{F}_F.append((q_i, \mathbf{f}_i))$ // Add identified bit positions to $\mathbf{F}_F$
  $\mathbf{r}_i^+, \mathbf{r}_i^- \leftarrow UpdateRowFilter(\mathbf{f}_i, \mathbf{B})$ // Based on bit positions in $\mathbf{f}_i$
  $\forall b \in \mathbf{r}_i^+ : \mathbf{A}_F^+[b] \leftarrow \mathbf{A}_F^+[b] \cup \{q_i\}$ // Assign q_i to must have q-gram sets
  $\forall b \in \mathbf{r}_i^- : \mathbf{A}_F^-[b] \leftarrow \mathbf{A}_F^-[b] \cup \{q_i\}$ // Assign q_i to cannot have q-gram sets
return $\mathbf{F}_F$, $\mathbf{A}_F^+$, $\mathbf{A}_F^-$ // Return final lists

Using the plaintext QID values we check whether if any of the q-grams in the subset does not co-occur with the other q-grams in the same subset. (i.e. the intersection of QID values of a q-gram with QID values of other q-grams is empty). If there are such q-grams in the considered subset we assign that subset as not valid since we have already processed a similar subset in a previous iteration as shown in the following example.

In our previous example of the frequent q-gram set $\{sm, mi, th\}$, consider the subsets of length $n = 3$, $\mathbf{M} = \{\{sm, mi, th\}\}$. Assume the q-gram '*sm*' from this subset does not occur with the other two q-grams, '*mi*' and '*th*', in any plaintext value,

i.e. $(\{\mathbf{q} \in \mathbf{Q} : \text{'}sm\text{'} \in \mathbf{q}\} \cap \{\mathbf{q} \in \mathbf{Q} : \text{'}mi\text{'} \in \mathbf{q}\} = \emptyset) \wedge (\{\mathbf{q} \in \mathbf{Q} : \text{'}sm\text{'} \in \mathbf{q}\} \cap \{\mathbf{q} \in \mathbf{Q} : \text{'}th\text{'} \in \mathbf{q}\} = \emptyset)$. In a such case we assign the subset $\{sm, mi, th\}$ as invalid since only '*mi*' and '*th*' are able to contribute to the partitioning step, and the subset $\{mi, th\}$ was previously processed when the subset length was $n = 2$.

In line 7, we then loop over each instance, $\mathbf{s} \in \mathbf{S}$, to generate the corresponding BF and plaintext partitions. In lines 8 to 10 the column filter $\mathbf{c}_s$, the row filter (BF partition) $\mathbf{r}_s$, and the plaintext partition $\mathbf{q}_s$ are initialised by assigning them all bit positions, BFs, and q-gram sets of plaintext values, respectively. Next, from lines 11 to 18, we loop over each bit value $s_i \in \mathbf{s}$ and each q-gram $q_i \in \mathbf{m}$, and update $\mathbf{c}_s$, $\mathbf{r}_s$, and $\mathbf{q}_s$ according to the bit value in s_i. If $s_i = 0$ (the q-gram q_i cannot be in the current instance), then we update $\mathbf{r}_s$ and $\mathbf{q}_s$ by removing all BF and q-gram sets corresponding to the q-gram q_i (lines 13 and 14). If $s_i = 1$ (the q-gram q_i must be in the current instance), then we update $\mathbf{r}_s$ and $\mathbf{q}_s$ by obtaining the intersection of BF and q-gram sets that correspond to the q-gram q_i (lines 16 and 17).

The column filter is updated in line 18 by removing the bit positions of the frequent q-gram q_i. In line 19 we then append the obtained BF and plaintext partitions to the queue Q along with the column and row filters. If the considered q-gram subset is not valid then we continue to the next subset. The algorithm terminates when all frequent q-gram subsets have been processed and added to the queue Q.

7.6.2 Identifying Further Q-grams using the Partitions

After generating the BF and plaintext partitions, in Algorithm 7.5 we aim to identify further q-grams and their co-occurring bit positions by applying pattern-mining on those partitions. This algorithm follows a similar structure as Algorithm 7.3. It starts by initialising the final lists $\mathbf{F}_F$, $\mathbf{A}_F^+$, and $\mathbf{A}_F^-$ of identified q-grams and must and cannot have q-gram sets. A new q-gram list $\mathbf{F}_N$ is initialised in line 2 to hold the initial results of the pattern-mining process.

The processing of partitions from lines 3 to 16 is similar to the process in Algorithm 7.3 from lines 9 to 24 (on page 142). The main loop to process Q starts in line 3 and runs until Q is empty. In contrast to Algorithm 7.3, we employ an error limit, k_{err}, for the estimated number of hash functions k_{est} since we might not always get exactly k_{est} co-occurring bit positions for a certain q-gram because of collisions with bit positions of the previously identified q-grams. In line 12 we use k_{err} to check if the length of $\mathbf{f}_i$ is within the defined range. If it is then we append $\mathbf{f}_i$ to $\mathbf{F}_N$ along with q_i in line 13. In line 14 we update the column filter $\mathbf{c}_s$ by removing the identified bit positions.

Since the algorithm processes a large number of partitions (depending on the number of frequent q-grams identified in the first step), there is a possibility that a certain q-gram q_i might be assigned to more than one maximal frequent bit position tuple. From lines 17 to 26 we therefore analyse each identified q-gram and its corresponding bit position tuple(s) to find the single most appropriate tuple for each q-gram. In line 17 we iterate over each $q_i \in \mathbf{F}_N$. If the q-gram has only one bit position tuple assigned to it then we retrieve this tuple in line 20. Otherwise, if the

Algorithm 7.6: Reidentify plaintext values in Bloom filters

Input : **B**: List of BFs from the sensitive encoded database
V: List of plaintext values from a public database
Q: List of q-gram sets, one per plaintext value in **V**
$\mathbf{F}_F$: Final list of identified q-grams and their sets of BF bit positions
$\mathbf{A}_F^+$: Final list of expanded must have q-gram sets assigned to BFs in **B**
$\mathbf{A}_F^-$: Final list of expanded cannot have q-gram sets assigned to BFs in **B**
n_m: Minimum number of required identified q-grams in a BF from **B**

Output: **R**: List of reidentified plaintext values from **V** for BFs from **B**

1 $\mathbf{I}_B \leftarrow [\,], \mathbf{R} \leftarrow [\,]$ // Initialise inverted index $\mathbf{I}_B$ and list **R**
2 $\mathbf{I}_Q \leftarrow GenValQGramInvIndex(\mathbf{V}, \mathbf{Q})$ // Create inverted index for unique q-grams
3 **for** $b \in \mathbf{A}_F^+$ **do**: // Loop over each BF in must have q-gram list
4 $\mathbf{q}^+ \leftarrow \mathbf{A}_F^+[b]$, $\mathbf{q}^- \leftarrow \mathbf{A}_F^-[b]$ // Get must have and cannot have q-gram sets
5 **for** $(q_i, \mathbf{f}_i) \in \mathbf{F}_F$ **do**: // Loop over each q-gram and its bit positions
6 **if** $\exists p \in \mathbf{f}_i : \mathbf{B}[b][p] \leftarrow 0$ **then**: // If any identified position in b is set to 0
7 $\mathbf{q}^- \leftarrow \mathbf{q}^- \cup \{q_i\}$ // Add q_i to cannot have q-gram set
8 $\mathbf{I}_B[(\mathbf{q}^+, \mathbf{q}^-)].add(b)$ // Add b to $(\mathbf{q}^+, \mathbf{q}^-)$ tuple in $\mathbf{I}_B$
9 **for** $(\mathbf{q}^+, \mathbf{q}^-) \in \mathbf{I}_B$ **do**: // Loop over each $(\mathbf{q}^+, \mathbf{q}^-)$ tuple in $\mathbf{I}_B$
10 **if** $|\mathbf{q}^+| \geq n_m \vee |\mathbf{I}_B[(\mathbf{q}^+, \mathbf{q}^-)]| \leftarrow 1$ **then**: // If size of $\mathbf{q}^+$ is above the threshold
11 $\mathbf{v} \leftarrow \{\forall q_i \in \mathbf{q}^+ : \cap \mathbf{I}_Q[q_i]\} \setminus \{\forall q_i \in \mathbf{q}^- : \cup \mathbf{I}_Q[q_i]\}$ // Get set of plaintext values
12 **for** $b \in \mathbf{I}_B[(\mathbf{q}^+, \mathbf{q}^-)]$ **do**: // Loop over each BF in current $(\mathbf{q}^+, \mathbf{q}^-)$ tuple
13 **for** $v \in \mathbf{v}$ **do**: // Loop over each plaintext value in **v**
14 $\mathbf{R}[b] \leftarrow \mathbf{R}[b] \cup \{v\}$ // Assign identified plaintext values to BFs
15 **return R**

q-gram q_i has more than one bit position tuple assigned to it then we first calculate the number of times each bit position occurs in all of those bit position tuples. In line 22, the *GetMostFreqBitPos*() calculates the set of most frequent positions in all the identified tuples, and outputs the k_{est} most frequent bit positions. Next, in line 23, we re-assign the new position tuple to the final list of identified q-grams, $\mathbf{F}_F$. The two lists of must have and cannot have q-gram sets for BFs in **B** are then updated in lines 25 and 26, respectively, using the updated row filters $\mathbf{r}^+$ and $\mathbf{r}^-$. Finally, in line 27 we return the final lists $\mathbf{F}_F$, $\mathbf{A}_F^+$, and $\mathbf{A}_F^-$.

7.7 Reidentification of Plaintext Values

As detailed in Algorithm 7.6 [216], the final step of our attack is to reidentify plaintext QID values that could have been encoded into BFs in **B**. We use the final lists of identified q-grams and their bit positions, $\mathbf{F}_F$, and the lists $\mathbf{A}_F^+$ and $\mathbf{A}_F^-$ of must have and cannot have q-gram sets for BFs, as generated in Algorithm 7.5 for the reidentification process. We only consider plaintext values and BFs that contain at least n_m must have q-grams, because considering a single or only a few identified q-grams would result in too many plaintext values that could match a BF. For example, 3,700 records from a total of 100,000 records in the experimental NCVR-100k database (as described in Section 7.9) have last names which contain the q-gram 'th'.

The algorithm starts by initialising an inverted index $\mathbf{I}_B$ which for each pair of must have and cannot have q-gram sets from $\mathbf{A}_F^+$ and $\mathbf{A}_F^-$ a list of all BFs in **B** that

contain all of the must have and none of the cannot have q-grams is assigned. In line 2 a second inverted index, $\mathbf{I}_Q$, is generated which for each unique q-gram (keys in $\mathbf{I}_Q$) all values $v \in \mathbf{V}$ that contain this q-gram are assigned. This inverted index is used to efficiently identify candidate sets of possible plaintext values that can be encoded in a certain BF.

In lines 3 to 8, for each BF b that has must have and cannot have q-grams assigned to it (in $\mathbf{A}_F^+$ and $\mathbf{A}_F^-$), we identify any other q-grams in $\mathbf{F}_F$ that cannot be encoded in this BF because at least one bit position assigned to such a q-gram in $\mathbf{f}_i$ is set to 0 in this BF. We expand the set of cannot have q-grams, $\mathbf{q}^-$, accordingly, and in line 8 add the BF b to the list of BFs in the inverted index $\mathbf{I}_B$ with the key $(\mathbf{q}^+, \mathbf{q}^-)$. In the loop in lines 9 to 14 we first check for each tuple $(\mathbf{q}^+, \mathbf{q}^-) \in \mathbf{I}_B$ if at least n_m q-grams are included in the must have q-gram set $\mathbf{q}^+$, or if only one BF is attached to the tuple. For those tuples fulfilling the condition, a set of possible values $\mathbf{v} \subset \mathbf{V}$ that contain all q-grams in $\mathbf{q}^+$ and no q-gram in $\mathbf{q}^-$ are generated in line 11. Using the inverted index $\mathbf{I}_Q$ this is accomplished by efficient set intersections and unions of plaintext values. Finally, in lines 12 to 14, we add each identified value $v \in \mathbf{v}$ to the set of reidentified values in the list $\mathbf{R}$ for each BF b with the given pair of must and cannot have q-gram sets, $(\mathbf{q}^+, \mathbf{q}^-)$. Finally, we return $\mathbf{R}$ in line 15.

7.7.1 Length Filtering Process

We finally propose a length filtering approach which we use to reduce the number of plaintext values assigned to each BF in $\mathbf{R}$ as returned by Algorithm 7.6. If a BF has several plaintext values assigned to it, reidentification becomes less precise. For each BF $b \in \mathbf{R}$ we calculate a lower and upper bound, n_q^l and n_q^u, for the number of q-grams that can be encoded in b. Using these bounds we then remove the plaintext values assigned to b where their number of q-grams are outside $[n_q^l, n_q^u]$.

For a given BF, b, with a Hamming weight of w_H (number of 1-bits), the lower bound can be calculated as $n_q^l = w_H / k_{est}$, where k_{est} is the estimated number of hash functions as calculated by the pattern-mining step in Algorithm 7.1. Assuming n_q^t is the true number of q-grams in the encoded plaintext value and there are no collisions in the BF b, then w_H can be calculated as $w_H = n_q^t \cdot k_{est}$. If there are collisions then $w_H < n_q^t \cdot k_{est}$, and therefore $(w_H / k_{est} = n_q^l) \leq n_q^t$ must always hold. To calculate the upper bound, n_q^u, we first calculate the expected number of q-grams encoded in this BF b. According to Swamidass and Baldi [188], the approximate number of elements (q-grams in our case) encoded into b can be calculated as $q_a = -(l/k_{est}) \cdot ln(1 - w_H/l)$, where l is the BF length. From this we calculate the upper bound q^u by using the symmetric bound, $n_q^u = q_a + (q_a - n_q^l)$.

After applying length filtering to the list $\mathbf{R}$ at the end of Algorithm 7.6, we return another list, $\mathbf{R}_F$, which contains only the filtered plaintext values $v \in \mathbf{V}$ for each BF b that have a number of q-grams $n_q^l \leq |\mathbf{q}| \leq n_q^u$. We report the accuracy of this length filtering step in Table 7.5.

7.8 Complexity Analysis

We evaluate the worst-case computational complexity of our attack by analysing the number of computational steps of each algorithm discussed in Sections 7.4 to 7.7. The complexity of Algorithm 7.1 is as follows. The initialisation steps in line 1 and 5 have complexity of $O(|\mathbf{B}|)$ whereas the initialisation step in line 3 has complexity of $O(|\mathbf{V}|)$. If in each iteration of the main loop k bit positions are assigned to a frequent q-gram (in line 15) then $O(l/k)$ iterations can be expected. If we assume the worst-case complexity of the pattern-mining algorithm as quadratic in the number of items (BF length l) and linear in the number of transactions (BFs) [85], then the overall complexity of the main loop becomes $O(l^3/k \cdot |\mathbf{B}|)$.

The complexity of Algorithm 7.2 is $O(|\mathbf{F}| \cdot |\mathbf{B}| \cdot n_q)$, where n_q is the average number of q-grams in the sets $\mathbf{q}_j \in \mathbf{Q}$. The complexity of Algorithm 7.3 is dominated by the number of iterations and the worst-case complexity of the pattern-mining algorithm, leading to an overall complexity of $O(l^3/k \cdot |\mathbf{B}|)$. Populating the queue Q (in lines 3 to 6) depends on the number of frequent q-grams, where for each q-gram a scan over all BFs is required to generate its row filter set, $\mathbf{r}_f$ (in line 4), resulting in a $O(|\mathbf{F}| \cdot |\mathbf{B}|)$ complexity.

The complexities of both Algorithms 4 and 5 are dominated by the number of BF and plaintext partition pairs processed. Algorithm 7.4 loops over each frequent q-gram subsets $\mathbf{m} \in \mathbf{M}$ of different lengths $n \in \mathbf{n}$ and generates partitions for each possible instance $\mathbf{s} \in \mathbf{S}$. Considering these functionalities the complexity of Algorithm 7.4 can be defined as $O(\sum_{n \in \mathbf{n}} \binom{|\mathbf{F}|}{n} \cdot 2^n)$, where n is the considered subset length, $\binom{|\mathbf{F}|}{n}$ is the number of subsets we obtain when choosing n q-grams from all frequent q-grams in $\mathbf{F}$, and 2^n corresponds to the number of instances each subset of length n contains. Since the complexity increases exponentially with n, it leads to an overall worst-case complexity of $O(2^n)$. However, due to the pruning step used in Algorithm 7.4 the true complexity is much lower than this in practice.

Since Algorithm 7.5 also loops over each partition pair in Q and then processes all q-gram pairs in each partition, its complexity can be defined as $O(|Q| \cdot \binom{|\mathbf{q}_f|}{2})$ where $|\mathbf{q}_f|$ is the number of other q-grams in each considered partition. The length of Q is the number of partitions generated in Algorithm 7.4. In the worst-case this length should therefore be the same as the complexity of Algorithm 7.4, $O(2^n)$. In practice, however, Algorithm 7.5 does not loop over every q-gram pair because many do not have the required minimum percentage difference, d. Therefore, the processed number of q-gram pairs for each partition can be lower than $|Q|$ which leads to an expected complexity of $O(|Q|)$ for Algorithm 7.5.

The complexity of generating the inverted index $\mathbf{I}_B$ in Algorithm 7.6 is $O(|\mathbf{B}| \cdot |\mathbf{F}_F|)$ because it loops over all BFs in $\mathbf{A}_F^+$ and for each checks all identified q-grams in $\mathbf{F}_F$. The second loop (from lines 9 to 14) will have $O(|\mathbf{V}| \cdot |\mathbf{I}_Q|)$ complexity assuming each value $v_j \in \mathbf{V}$ is unique and thus $|\mathbf{I}_B| \leq |\mathbf{V}|$, where we assume a linear time for a set intersection and a set union, and that $|\mathbf{I}_Q|$ corresponds to the number of unique q-grams from $\mathbf{V}$.

7.9 Experimental Evaluation

We evaluated our proposed pattern-mining based attack method on BF encoding using different instances of a real-world database. We evaluated different parameter settings as well as different attack scenarios to measure the feasibility of our attack in real-world linkage scenarios.

7.9.1 Databases and Experimental Setup

As we discussed in Chapter 6, we used a snapshot of the North Carolina Voter Registration (NCVR) database (http://dl.ncsbe.gov/data/) to create 20 sensitive ($\mathbf{D}^s$) and plaintext ($\mathbf{D}^p$) database pairs. As mentioned, there were five sets of database pairs containing 20,000 (20k), 100,000 (100k), 200,000 (200k), 500,000 (500k), and 1,000,000 (1M) records, where in each set we had four database pairs which had 100%, 75%, 50%, and 0% matching records between them.

Using both the Cryptographic Long-term Key (CLK) [178] and Record-level BF (RBF) [61] approaches (as we discuss in Appendix A) we encoded different combinations of the attributes *FirstName*, *LastName*, *StreetAddress*, and *City* into BFs. For combinations of three and four attributes in the NCVR-20k, NCVR-100k, and NCVR-200k database pairs, all values in $\mathbf{V}$ and all bit patterns in $\mathbf{B}$ were unique. Following earlier work [27, 61, 64, 177], we used the BF encoding settings: $l_q = 2$, $l = 1,000$, random hashing [180], and $k = [10, 15, 30, opt]$ where opt is the optimal number of hash functions such that the average number of 1-bits in a BF is 50% to minimise the collisions in BFs [205]. We set the values for the minimum percentage difference $d = [1.0, 5.0]$ in Algorithms 1, 2, and 3, respectively; the minimum partition size $m = [2,000, 10,000]$ in Algorithm 7.1; the minimum q-gram tuple size $n_m = 3$ in Algorithm 7.6; the lengths to create q-gram combinations $\mathbf{n} = [2, 3]$ in Algorithm 7.4; and the error limit for the estimated number of hash functions $k_{err} = 2$ in Algorithm 7.5. For the refinement step discussed in Section 7.4.2, we set the value for the maximum percentage difference $\theta = 5\%$ in Lemma 7.4.1, the lower conditional probability limit $\alpha = 0.1$ in Lemma 7.4.2, and the confidence threshold $\beta = 0.95$ in Lemma 7.4.2. These settings provided good results in a series of set-up experiments.

7.9.2 Evaluation Criteria

We evaluated the quality of the identified frequent q-grams in $\mathbf{F}$, $\mathbf{F}_E$, and $\mathbf{F}_F$ from Algorithms 1, 3, and 5 as the precision and recall of how many bit positions were correctly identified for a q-gram in $\mathbf{F}$, $\mathbf{F}_E$, or $\mathbf{F}_F$, averaged over all q-grams. We evaluated the precision and recall for different numbers of hash functions, k, and different attribute combinations. We measured the accuracy of the estimated number of hash functions k_{est} by comparing those to the actual number of hash functions used, and the length filtering step discussed in Section 7.7.1 by calculating the percentage of plaintext QID values falls within the lower(n_q^l) and upper bounds (n_q^u).

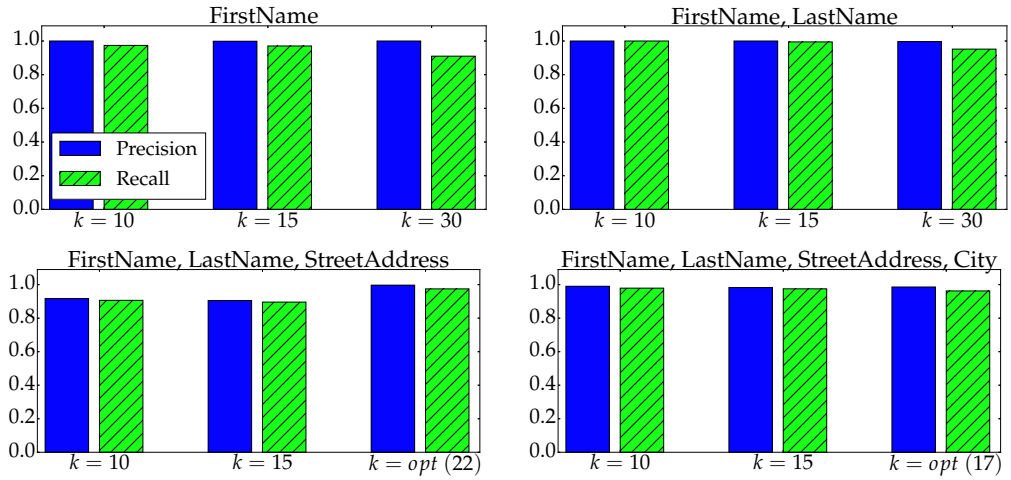
Table 7.5: Summary results of the proposed attack for different attribute combinations analysed using the NCVR-100k databases. The first two columns show the true and estimated numbers of hash functions, k and k_{est}. The next three columns show the minimum and maximum numbers of identified q-grams in $|\mathbf{F}|$, $|\mathbf{F}_E|$, and $|\mathbf{F}_F|$, respectively. The fifth and sixth columns show the percentages of plaintext values where their true number of q-grams, n_q^t, is within the lower (n_q^l) and upper (n_q^u) bounds calculated in the length filtering step in Section 7.7.1. The last column shows the average reduction ratio, $RR = 1 - (|\mathbf{R}_l[b]|/|\mathbf{R}[b]|)$, of the numbers of reidentified plaintext values assigned to BFs when the length filtering step is applied, where $\mathbf{R}[b]$ is the number of plaintext values assigned to a BF b before filtering and $\mathbf{R}_l[b]$ is the number of plaintext values assigned to the same BF after length filtering.

Attribute combination	k	k_{est}	$\lvert\mathbf{F}\rvert$	$\lvert\mathbf{F}_E\rvert$	$\lvert\mathbf{F}_F\rvert$	$n_q^t \geqslant n_q^l$	$n_q^t \leqslant n_q^u$	RR
FirstName	10	10	10-20	18-33	31-84	100%	99.9%	0.28
	15	15	10-20	18-33	30-83	100%	99.9%	0.29
	30	26/28	10-14	19-26	25-37	100%	99.9%	0.32
FirstName and *LastName*	10	10	7-13	12-20	36-62	100%	99.9%	0.28
	15	15	7-13	12-20	33-41	100%	99.9%	0.28
	30	27/29	7-13	12-20	35-61	100%	100%	0.19
FirstName,	10	10	6-12	8-16	20-46	100%	99.9%	0.29
LastName, and	15	15	6-12	8-15	21-46	100%	100%	0.17
StreetAddress	22 (opt)	22	5-10	6-12	13-24	100%	100%	0.08
FirstName,	10	10	14-15	19-20	31-45	100%	100%	0.03
LastName, *Street-*	15	15	8-10	12-15	24-29	100%	100%	0.03
Address, and *City*	17 (opt)	16/17	8-9	11-12	17-18	100%	100%	0.05

In terms of reidentifications we evaluated the accuracy of the reidentified plaintext QID values for BFs and reidentified entities using the number of 1-to-1, 1-to-*many*, *many*-to-1, and *many*-to-*many* reidentifications as we discussed in Section 3.7.1 on page 55. We also calculated the probability of suspicion (PS) values for each record in the encoded database $\mathbf{D}^e$ as we discussed in Section 3.7.1.

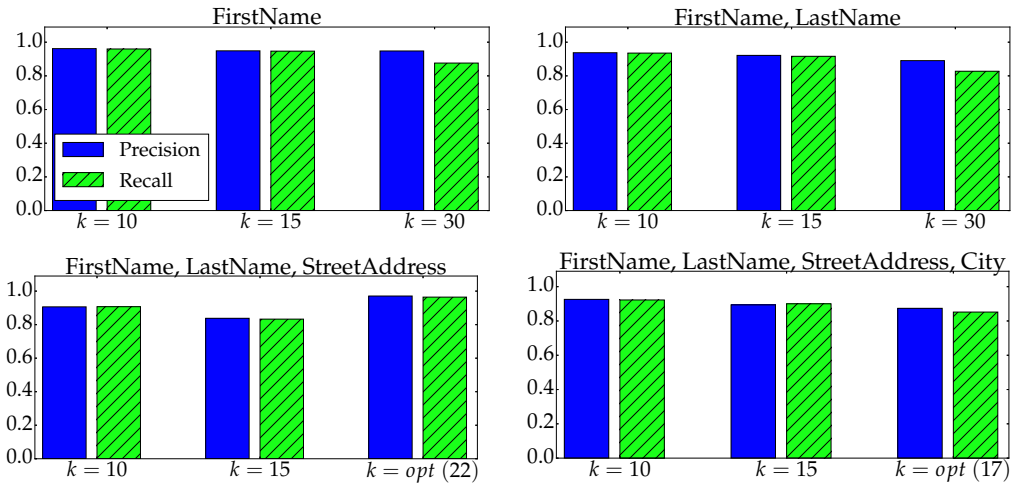
7.9.3 Results and Discussion

In Table 7.5 we summarise the results of our attack method using where the third column shows the accuracy of estimating the number of hash functions, k_{est} and the fourth, fifth, and sixth columns show the the number of q-grams identified in the first three steps of the attack, $\mathbf{F}$, $\mathbf{F}_E$, and $\mathbf{F}_F$. As can be seen, our attack method can correctly identify the number of hash functions, k, across almost every experiment. Furthermore, our attack can identify more q-grams in the novel second and third steps (in $\mathbf{F}_E$ and $\mathbf{F}_F$) compared to the first step (in $\mathbf{F}$), with an increment in the number of identified q-grams ranging from 33% to over 400%. These identified additional q-grams will allow more attribute values encoded in BFs in $\mathbf{B}$ to be reidentified.



(i) Precision and recall results of q-grams in $\mathbf{F}$ (Step 1) from Algorithm 7.1.

(ii) Precision and recall results of q-grams in $\mathbf{F}_E$ (Step 2) from Algorithm 7.3.



(iii) Precision and recall results of q-grams in $\mathbf{F}_F$ (Step 3) from Algorithm 7.5.

Figure 7.4: Precision and recall results of identified q-grams in the first three steps of the attack on the NCVR-100k databases. The results are presented for different QID attribute combinations and different numbers of hash functions, k.

Table 7.6: Attribute reidentification accuracy results for the NCVR-100k database pairs with different number of matching records. We present the results for all four attribute combinations.

Attribute combination	Percentage of matches	Christen et al. [43] / Our new attack method				
		1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FirstName	100%	22 / 2,071	0 / 0	41 / 1,458	0 / 0	0 / 235
	75%	15 / 1,149	0 / 0	45 / 1,036	0 / 0	11 / 333
	50%	7 / 561	0 / 0	14 / 749	0 / 0	9 / 535
	0%	0 / 75	0 / 0	7 / 387	0 / 0	16 / 640
FirstName and *LastName*	100%	8 / 27,596	0 / 0	80 / 9,015	0 / 0	0 / 3,212
	75%	11 / 7,078	0 / 0	112 / 2,747	0 / 0	38 / 1,844
	50%	0 / 6,140	0 / 0	2 / 2,672	0 / 0	3 / 3,207
	0%	1 / 6,140	0 / 0	0 / 2,672	0 / 0	153 / 3,207
FirstName, *LastName*, and *StreetAddress*	100%	0 / 11,107	0 / 0	42 / 6,652	0 / 0	0 / 3,688
	75%	0 / 2,600	0 / 0	2 / 2,532	0 / 0	3 / 3,951
	50%	0 / 5,819	0 / 0	21 / 3,489	0 / 0	21 / 7,673
	0%	0 / 0	0 / 0	0 / 0	0 / 0	42 / 322
FirstName, *LastName*, *StreetAddress*, and *City*	100%	2,117 / 21,910	0 / 0	6,086 / 4,161	0 / 0	358 / 3,298
	75%	2,294 / 16,987	0 / 0	5,347 / 3,029	0 / 0	2,605 / 2,323
	50%	921 / 5,523	0 / 0	3,175 / 1,321	0 / 0	3,678 / 3,184
	0%	No frequent bit patterns found				

The last three columns in Table 7.5 show the percentages of plaintext values that fall within the calculated lower and upper bounds (as we discussed in Section 7.7.1) and the amount of reduction in candidate plaintext values assigned to BFs using the upper and lower bounds of the numbers of expected q-grams. According to these results, the length filtering process is highly accurate and also efficient in reducing many incorrectly reidentified plaintext values.

In Figure 7.4 we show the precision and recall results of the q-grams identified in the first three steps of the attack ($\mathbf{F}$, $\mathbf{F}_E$, and $\mathbf{F}_F$) using different number of hash functions and attribute combinations. These results are averaged over different database pairs and other parameter settings discussed in Section 7.9.1. As can be seen, both precision and recall of the identified frequent q-grams in $\mathbf{F}$ are above 0.8, for different values of k and attribute combinations. The results on all the database pairs with three and four attribute combinations validate that our attack can successfully identify bit positions of frequent q-grams with high accuracy even when each BF in an encoded database is unique.

However, for the identified bit positions of q-grams in $\mathbf{F}_E$, while the precision stays above 0.8, the recall of those bit positions drops. This is due to collisions in bit positions by multiple q-grams, as we discussed in Section 7.5.4. To identify bit positions of q-grams more accurately, we employed the partitioning method discussed in Section 7.6. As the results for $\mathbf{F}_F$ in Figure 7.4 show, after the application of the partitioning method, both precision and recall of the identified q-grams in $\mathbf{F}_F$ stay above 0.85, for most attribute combinations and numbers of hash functions.

Table 7.7: Identity reidentification accuracy results for the NCVR-100k database pairs with different number of matching records. We present the results for all four attribute combinations.

Attribute combination	Percentage of matches	Christen et al. [43] / Our new attack method				
		1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FirstName	100%	22 / 1,934	0 / 23	0 / 74	41 / 1,257	0 / 20
	75%	14 / 998	3 / 101	3 / 157	50 / 860	1 / 103
	50%	4 / 413	3 / 84	1 / 88	19 / 643	3 / 194
	0%	0 / 0	0 / 0	0 / 0	0 / 0	24 / 769
FirstName and *LastName*	100%	8 / 25,646	0 / 492	0 / 615	80 / 8,229	0 / 311
	75%	10 / 6,617	3 / 430	2 / 1,061	142 / 1,935	4 / 724
	50%	0 / 5,557	0 / 483	0 / 1,169	5 / 1,900	0 / 1,828
	0%	0 / 0	0 / 0	0 / 0	0 / 0	154 / 5,091
FirstName, *LastName*, and *StreetAddress*	100%	0 / 9,830	0 / 368	0 / 268	42 / 6,275	0 / 40
	75%	0 / 2,251	0 / 350	0 / 770	5 / 2,392	0 / 1,208
	50%	0 / 5,006	0 / 1,033	0 / 1,072	42 / 4,078	0 / 2,504
	0%	0 / 0	0 / 0	0 / 0	0 / 0	42 / 290
FirstName, *LastName*, *StreetAddress*, and *City*	100%	2,033 / 21,328	201 / 683	35 / 893	6,292 / 3,747	0 / 1,698
	75%	1,981 / 16,370	767 / 829	502 / 1,422	6,639 / 2,022	357 / 1,074
	50%	708 / 5,328	457 / 353	369 / 674	5,481 / 1,306	759 / 1,888
	0%	No frequent bit patterns found				

Tables 7.6 and 7.7 show the attribute and identity reidentification results, respectively, from the fourth step of our attack as discussed in Algorithm 7.6. Compared to the initial version of this attack [43], the improved version proposed in this chapter was able to reidentify a substantially larger number of plaintext QID values and entities. When comparing 1-to-1 matches in attribute reidentification our new attack method gained a minimum increment of around 5-fold and a maximum increment of nearly 3,500-fold (from 8 to 27,596 reidentifications) compared to our basic attack. Similarly, when comparing 1-to-1 matches in identity reidentification our new attack method gained a minimum increment of around 6-fold and a maximum increment of nearly 3,200-fold (from 8 to 25,646 reidentifications). Interestingly, with increasing number of encoded attributes the attack was able to reidentify more entities. This is because with the combination of more attributes, the records in a database becomes more unique and therefore the entities can be uniquely identified. However, for the database pair with four attribute combination and 0% matching records, the attack was not able to identify frequent bit patterns in the encoded database that corresponds to frequent q-grams in the plaintext database. This is because of the considerably large differences in QID attribute values between databases due to 0% matching records.

Similar results can also be observed in the Table 7.8 where we show the PS values for the reidentification of entities. All of the of the databases and attribute combinations had a maximum risk of 1.0. A minimum risk value 0 indicates that not all records in the encoded database are assigned with plaintext QID values and therefore

Table 7.8: Maximum, minimum, mean, median, and marketer risk values of the records in the NCVR-100k encoded sensitive databases calculated based on probability of suspicion (PS) as we discussed in Section 3.7.1 on page 55.

Attribute combination	Percentage of matches	Christen et al. [43] / Our new attack method				
		Maximum risk	Minimum risk	Mean risk	Median risk	Marketer risk
FirstName	100%	1.0 / 1.0	0 / 0	0.02 / 0.08	0 / 0.01	0.001 / 0.021
	75%	1.0 / 1.0	0 / 0	0.02 / 0.07	0 / 0.01	< 0.001 / 0.013
	50%	1.0 / 1.0	0 / 0	0.01 / 0.05	0 / 0.01	< 0.001 / 0.007
	0%	0 / 0	0 / 0	0.01 / 0.03	0 / 0.01	< 0.001 / 0.001
FirstName and *LastName*	100%	1.0 / 1.0	0 / 0	0.00 / 0.53	0 / 0.57	< 0.001 / 0.28
	75%	1.0 / 1.0	0 / 0	0.02 / 0.69	0.01 / 0.83	0.002 / 0.094
	50%	1.0 / 1.0	0 / 0	0.01 / 0.64	0 / 0.78	< 0.001 / 0.093
	0%	0 / 0	0 / 0	0.00 / 0.37	0 / 0.24	< 0.001 / 0.026
FirstName, *LastName*, and *StreetAddress*	100%	1.0 / 1.0	0 / 0	0.00 / 0.24	0 / 0.10	< 0.001 / 0.109
	75%	1.0 / 1.0	0 / 0	0.00 / 0.25	0 / 0.11	< 0.001 / 0.034
	50%	1.0 / 1.0	0 / 0	0.01 / 0.43	0 / 0.41	< 0.001 / 0.067
	0%	0 / 0	0 / 0	0.01 / 0.61	0 / 0.50	< 0.001 / < 0.001
FirstName, *LastName*, *StreetAddress*, and *City*	100%	1.0 / 1.0	0 / 0	0.03 / 0.66	0.01 / 0.71	0.007 / 0.233
	75%	1.0 / 1.0	0 / 0	0.11 / 0.83	0.04 / 1.0	0.027 / 0.194
	50%	1.0 / 1.0	0 / 0	0.03 / 0.65	0.01 / 0.69	0.004 / 0.073
	0%	No frequent bit patterns found				

these records had no identity reidentifications. A maximum risk of 1.0 for database pairs with 0 matches indicate that our attack was able to uniquely match pairs of records from encoded and plaintext databases based on the reidentified QID values even though those records belong to different real-world entities.

In Table 7.9, we show a summary of the accuracy of the identified q-grams with wrong bit positions as discussed in Section 7.4.2, and the improvements in reidentifications we gained by applying the length filtering step proposed in Section 7.7.1. We measured the precision and recall of identified q-grams with wrong bit positions, $\mathbf{w}_q$ (from Lemma 7.4.1) and the precision and recall of identified wrong bit positions, $\mathbf{w}_p$ (from Lemma 7.4.2). As can be seen, we were able to identify q-grams with wrong bit positions with high accuracy. Both precision and recall of the identified q-grams are always at least 0.83. However, while the precision of the identified wrong bit positions stays relatively high, the recall of the identified wrong bit positions drops in most cases. This is because these bit positions have high Hamming weights due to the large number of collisions by different q-grams. As discussed in Lemma 7.4.2, the identification of wrong bit positions, $\mathbf{w}_p$ in Table 7.9, depends on the Hamming weight of these bit positions. Therefore, the recall of identified wrong bit positions drops while the precision remains high. Especially when an optimal (*opt* in Table 7.9) number of hash functions is used for three and four attribute combinations, the attack could not identify any of the wrong bit positions.

The increment percentages shown in the last two columns of Table 7.9 when length filtering is applied indicate that the attack was able to increase the numbers of exact 1-to-1 matches in the final reidentification step (as shown in Algorithm 7.6).

Table 7.9: Summary results of the proposed refinement steps discussed in Sections 7.4.2 and 7.7.1 for different attribute combinations and numbers of hash functions, k. The first two columns show the attribute combination and the number of hash function used. The second and third columns show the precision and recall results for the identified sets of q-grams with wrong bit positions, $\mathbf{w}_q$, from Lemma 7.4.1, and the identified sets of wrong bit positions, $\mathbf{w}_p$, from Lemma 7.4.2. We measured precision of $\mathbf{w}_q$ as how many of the identified q-grams actually contain wrong bit positions, and recall as how many q-grams with wrong positions identified from all the q-grams with wrong bit positions. Similar calculations are used for the precision and recall of identified wrong bit positions, $\mathbf{w}_p$. Note that for the experiments with three and four encoded attributes and the opt number of hash functions the attack could not identify any of the wrong bit positions. Finally, the last two columns show the increment percentages of 1-to-1 and *many*-to-1 reidentifications, $\mathbf{R}_{inc()}$, after applying length filtering (Section 7.7.1) in Algorithm 7.6.

Attribute combination	k	$\mathbf{w}_q$ Precision/Recall	$\mathbf{w}_p$ Precision/Recall	$\mathbf{R}_{inc(1-1)}$	$\mathbf{R}_{inc(many-1)}$
FirstName	10	0.99 / 0.98	1.0 / 0.28	61.8%	-10.9%
	15	0.99 / 0.95	1.0 /0.1	50.3%	-12.2%
	30	1.0 / 1.0	1.0 / 0.2	83.7%	37.0%
FirstName and *LastName*	10	1.0 / 1.0	1.0 / 0.23	18.4%	-13.9%
	15	1.0 / 0.99	1.0 / 0.14	16.1%	-6.0%
	30	1.0 / 0.98	1.0 / 0.15	10.9%	2.2%
FirstName, *LastName*, and *StreetAddress*	10	1.0 / 0.87	1.0 / 0.04	17.4%	7.0%
	15	1.0 / 0.94	0.87 / 0.05	6.5%	1.8%
	opt=22	1.0 / 0.93	– / –	11.7%	6.8%
FirstName, *LastName*, *StreetAddress*, and *City*	10	1.0 / 0.83	1.0 / 0.24	1.5%	-12.5%
	15	1.0 / 0.84	1.0 / 0.04	2.1%	-3.9%
	opt=17	1.0 / 0.87	– / –	4.7%	3.0%

The negative values in the exact *many*-to-1 increment column indicate that most of the *many*-to-1 match results before the application of length filtering became 1-to-1 matches after length filtering was applied, thus resulting in reduced numbers of exact *many*-to-1 matches.

Finally, in Table 7.10 we show the runtime and memory consumption results for different steps of our proposed attack. As can be seen, with increasing number of records in databases the attack consumes more time and memory. The plaintext reidentification step is the most time and memory consuming step which took a maximum of around 43 hours and consumed maximum of around 81GB memory while language model based expansion step is the least time and memory consuming step out of all steps. This is because with the language models the attack was able to identify a smaller number of q-grams and their corresponding bit positions compared to first and third steps. For the database with 1 million records we terminated the attack after two weeks of running and consuming more than 200GB of memory.

Table 7.10: Time (in seconds) and memory (in Megabytes) consumption analysis of different steps of the proposed attack for different databases averaged over other parameter settings. For each step we refer to its corresponding section where we describe that step.

	Time (sec) / Memory (MB)			
Number of records	Identify freq. q-grams Section 7.4	Language model expansion Section 7.5	Partitioning based expansion Section 7.6	Plaintext reidentification Section 7.7
20k	4 / 1,235	2 / 1247	222 / 1311	144 / 1,946
100k	5 / 1,521	7 / 1,614	1,320 / 1923	355.5 / 7,049
200k	105 / 7,898	37 / 8,002	7,378 / 8,365	9,402 / 57,562
500k	372 / 18,365	123 / 20,586	15,286 / 22,654	156,050 / 81,648
1M	Experiment terminated due to high memory consumption			

The experimental results of our proposed cryptanalysis attack illustrate potential weaknesses of BF encoding, and how vulnerable BFs can be when certain parameter settings are used for the encoding of QID values. Given that BF encoding is now being established in several real-world PPRL applications [10, 29, 154, 164], it is important to highlight the need for improved BF encoding techniques to prevent this type of attack.

7.9.4 Recommendations

Based on our observations and the experimental results we obtained for the proposed attack method on BF encodings, we now make the following recommendations to mitigate the risks of potential reidentification of QID values encoded in BFs.

- We recommend that owners of sensitive databases use our attack method to assess the vulnerabilities of their BF encoding technique and used parameter settings. This will allow the them to evaluate the privacy protection provided by an encoded BF database before it is being used in a real-world application, and to take counter measures to strengthen the encoding method so that this type of an attack can be prevented.

- Apply BF hardening techniques such as salting [178], Wolfram's rule 90 [182], BLoom-and-flIP (BLIP) [180, 199], Markov chaining [183], window based XORing [161], or re-sampling based XORing [161]. These techniques will distort the bit patterns in BFs that correspond to q-grams and improve the privacy of encoded databases. However, as shown in one of the previously proposed cryptanalysis attacks on BFs [41], some hardening techniques, such as BF balancing [180] and XOR folding [181], are also vulnerable to certain types of attacks.

- Employ different encoding parameters on BFs to make the bit patterns less frequent and more distinct from each other by using different secret keys [147]

with keyed-hash message authentication codes [19], different numbers of hash functions, different length q-grams, and even different hash functions for different attributes. This will lead to different bit positions for the same q-gram occurring in different attributes.

While the above recommendations will improve the privacy of encoded databases against our proposed cryptanalysis attack, there is no guarantee (as with any cryptographic technique) that those recommendations will ensure resilience against any future attack methods [138]. Furthermore, it is important to note that using most of the hardening technique mentioned above will have a trade-off between privacy and the final linkage quality obtained [180, 182]. Therefore, further research into improved and advanced encoding techniques for PPRL is required to ensure the privacy of sensitive data while keeping a good linkage quality.

7.10 Summary

We have presented a novel cryptanalysis attack on BF encoding which employs pattern-mining, a probabilistic language model, and a partitioning based frequency analysis to successfully reidentify q-grams and plaintext QID values encoded in BFs. Our attack exploits the frequencies of q-grams in QID values rather than the frequencies of QID values themselves. Therefore, the attack works even when all BFs in an encoded database are unique, making the attack feasible in real-world scenarios. Due to the application of BFs in real-world PPRL systems, it is important to investigate the limitations that are associated with BF encoding. Our attack will help database owners to identify potential privacy risks associated in their encoded databases which allows them to put necessary counter measures to mitigate these risks before using encoded BFs in any further analysis of linkage scenarios.

Chapter 8

A Graph Matching Attack on Privacy-Preserving Record Linkage

All except one of the existing attack methods on privacy-preserving record linkage (PPRL) we discussed in Chapter 4 are proposed for Bloom filter encoding. In this chapter we present a novel attack method based on graph matching that, unlike the previous attacks, is independent of the PPRL encoding technique used. In Section 8.1 we first provide a brief introduction to our attack method and a two recently proposed encoding techniques for PPRL. In Section 8.2 we then describe our contribution of this chapter. In Section 8.3 we describe the main steps of our attack in detail, and in Section 8.4 we discuss the impact of adjusting similarities of plaintext and encoded values in our attack method. In Section 8.5 we analyse the computational complexity of our attack, and in Section 8.6 we experimentally evaluate our proposed attack using a real-world database. Finally, in Section 8.7 we summarise our findings.

8.1 Introduction

As we discussed in Chapters 2 and 4, research has shown that Bloom filter (BF) encoding is susceptible to privacy attacks [41, 119, 121, 216]. These attacks including the attack we proposed in Chapter 7 exploits frequently occurring patterns in BFs to reidentify encoded values. To overcome these privacy weaknesses of BF encoding, alternative perturbation based encoding techniques for PPRL have been developed. One of these uses tabulation Min-hashing [186] while another recently proposed method uses two hash encoding steps [158] as we discuss below.

Tabulation Min-hash Encoding (TMH): TMH method for PPRL was proposed by Smith [186] which uses more complex hashing which makes it more secure than BF encoding. The basic idea behind this method is to use tabulation based hashing to perform min-hash based locality sensitive hashing.

Similar to BF encoding, in TMH encoding one bit array (of 0s and 1s) of length l is generated for each record in the sensitive database to be encoded. First l sets of look-up tables are created, where each set consists of c look-up tables. Each look-up table contains keys of length k with a key space of size 2^k, where $1 < k$, and each

key points to a random bit string. Each element in a set **s** is then hashed using a one-way hash function (such as MD5 or SHA2 [175]), and a fixed length binary value is extracted from such hash values. This binary value is divided into c sub-keys of length k, and each sub-key is used as the index into one of the previously generated look-up table to obtain a random bit string. c such random bit strings are retrieved using c sub-keys, and these bit strings are then XORed to generate a single bit string for each of the l positions.

For each $s \in \mathbf{s}$ such a bit string is generated using this table look-up mechanism, and the minimum of all those bit strings is then selected as a min-hash signature. Since there are l sets of look-up tables, for a given set **s**, l min-hash signatures are generated. Smith argued that generating tabulation min-hash signatures alone will not be able to provide enough security against privacy attacks [186]. Therefore, a 1-bit hashing mechanism is used to prevent against attacks where only the least significant bit of each of the previously generated l min-hash signatures is kept. These l bits are then concatenated into the final bit array that is used as the encoding of a sensitive value that is represented by the set **s** of q-grams.

This tabulation hashing based method provides improved security against privacy attacks but at the cost of much higher computational requirements. Generating l sets of look-up tables will increase both the time and memory consumption of the overall encoding process compared to BF encoding.

Two-Step Hash Encoding (2SH): This recently proposed novel encoding technique for PPRL aims to address both the privacy issues of BF encoding, as well as the high computational costs of TMH encoding [158]. The 2SH method employs two efficient hashing steps based on bit vectors and an integer mapping to provide accurate Jaccard similarity calculations with improved privacy protection.

Similar to BF encoding, the 2SH encoding process converts the quasi-identifying (QID) attribute values selected from each record into a set of q-grams **s**. Next, k independent hash functions, where $k > 1$, are employed to hash each q-gram $s \in \mathbf{s}$ into k bit vectors of length l. Unlike BF encoding, the 2SH encoding process generates one bit vector per hash function, resulting in k bit vectors each of length l. This corresponds to a bit matrix with k rows and l columns at the end of the first hashing step. In the second step of 2SH, each column of this matrix that contains at least one 1-bit is hashed by mapping the column bit vector into an integer value in a defined range. The column number along with a secret key is used in the hashing process such that two columns with the same bit pattern will not result in the same integer value. For hashing each column bit vector into an integer value, either a hash function such as SHA2 [175] or a pseudo-random number generator [175] can be employed. Columns that only contain 0-bits will not be hashed because they do not encode any q-grams, and 0-bit columns that are common in two bit matrices would result in additional common integer values in their corresponding encodings that would lead to incorrect similarity calculations.

At the end of this two-step hashing process each record from a sensitive database will be represented by a set of integers, and these sets can be used to calculate the Jaccard similarity [38] between pairs of encoded records.

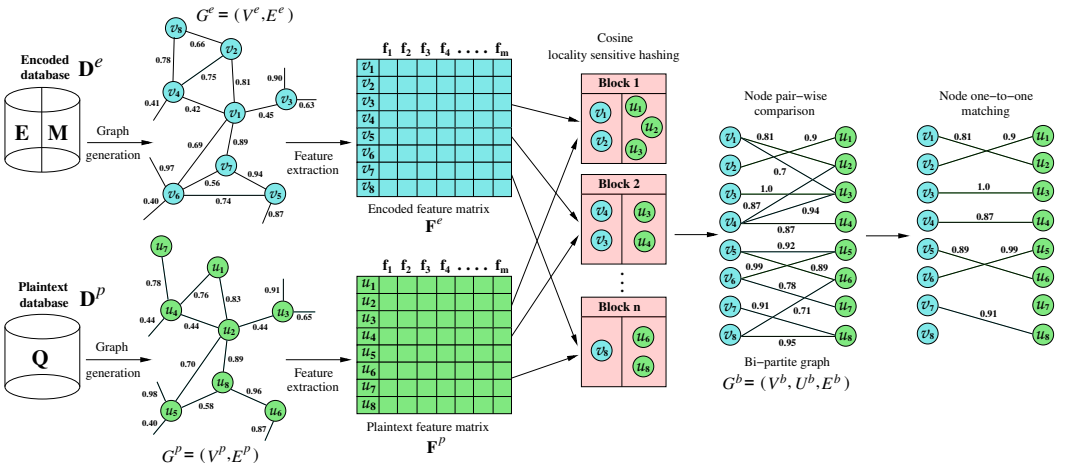


Figure 8.1: An overview of our proposed graph matching based attack. As we describe in Section 8.3, in the first step we generate two similarity graphs, G^e and G^p, from the encoded and plaintext databases, **E** and **Q**, respectively. Next we extract a set of features, **f**, for each node in both graphs using the node's neighbourhood, resulting in the two feature matrices, $\mathbf{F}^e$ and $\mathbf{F}^p$. We then use locality sensitive hashing (LSH) to identify groups of similar feature vectors in $\mathbf{F}^e$ and $\mathbf{F}^p$, and build a bi-partite graph connecting nodes $v_i \in \mathbf{F}^e$ with nodes $u_j \in \mathbf{F}^p$ based on their similar feature vectors. Finally, we apply one-to-one matching to identify which node v_i from **E** corresponds to which node u_j from **Q**, allowing us to reidentify the encoded value in v_i from u_j.

Given these two encoding techniques for PPRL are potentially to be used in real-world linkage applications where sensitive databases that contain personal information about people are being linked, it is important to understand their limitations and weaknesses in terms of the privacy protection they provide.

8.2 Contribution

While the majority of attacks on PPRL are limited to BF encoding, in this chapter we propose a novel attack method that exploits the calculated similarities between encoded records using a graph matching based approach. As illustrated in Figure 8.1, we generate one similarity graph from the encoded values and a second similarity graph from a set of plaintext values, where graph nodes correspond to the encoded or plaintext values, respectively, and graph edges correspond to the approximate similarities calculated between those values. We then apply graph matching techniques [3, 72, 91] to align nodes across these two graphs that are similar with regard to their local network structure, which allows us to match encoded values to plaintext values. This proposed attack can be applied on any PPRL encoding technique that calculates similarities between encoded values.

We investigate different scenarios of the attack, such as different assumptions about the resources an adversary might have access to, in order to understand the

Table 8.1: Specific notation used throughout this chapter. For common notation used throughout the thesis, refer to the list of notation at the beginning of the thesis.

$\mathbf{Q}$	List of plaintext quasi-identifying (QID) attribute values from $\mathbf{D}^p$
$\mathbf{F}^e$, $\mathbf{F}^p$	Encoded and plaintext feature matrix
$G^e = (V^e, E^e)$	Encoded similarity graph
$G^p = (V^p, E^p)$	Plaintext similarity graph
$G^b = (V^b, U^b, E^b)$	Encoded/plaintext bi-partite graph
$\mathcal{H}$	LSH family of hash functions
$N(v)$	Set of neighbouring nodes of a node v
$E(v)$	Set of edges connecting the neighbouring nodes in $N(v)$
$\mathbf{q}$	Unique set of q-grams extracted for a QID value
$\mathbf{f}$	Feature vector of a node
$\mathbf{c}$	Hyperplane that follow a uniform stable distribution
$v \in \mathbf{F}^e$, $u \in \mathbf{F}^p$	Node in $\mathbf{F}^e$ and a node in $\mathbf{F}^p$
s_m	Minimum similarity threshold considered in the graph generation step
c_m	Minimum connected component size considered in the feature generation step
$cs(u, v)$	Cosine similarity of an edge connecting the nodes u and v
$sc(u, v)$	Similarity confidence of an edge connecting the nodes u and v
$dc(u, v)$	Degree confidence of an edge connecting the nodes u and v
w_{cs}, w_{sc}, w_{dc}	Weight values for $cs(u, v)$, $sc(u, v)$, and $dc(u, v)$ to calculate a weighted edge value

strength and limitations of our attack on different PPRL encoding techniques. Our experimental results show that in certain situations these encoding techniques can be vulnerable to such a graph matching based attack where encoded values can be reidentified with high accuracy.

We specifically contribute (1) a novel attack using graph matching that works on any PPRL encoding techniques that calculates similarities between encoded values; (2) a method to adjust similarities between the encoded and plaintext graphs to improve the reidentification accuracy of our attack; and (3) an experimental evaluation using a real-world database and three perturbation based PPRL techniques (BF encoding and two recent techniques that aim to overcome the vulnerabilities of BFs).

8.3 A Graph Matching based Privacy Attack

We now describe the methodology of our graph matching based attack method on PPRL, as illustrated in Figure 8.1. We first discuss the preliminaries and assumptions of our attack and then describe the four main steps of the attack in detail. Specific notation we use throughout this chapter are shown in Table 8.1.

8.3.1 Preliminaries and Assumptions

We assume a linkage unit (LU) [209] is employed to conduct the linkage of the encoded databases it receives from the database owners (DOs). The LU can therefore build a similarity graph from the encoded values it is comparing. We assume the LU follows the honest-but-curious adversarial model [209], where it is curious to learn about the sensitive information in these encoded databases (the LU can be the adversary in our attack). We also assume the LU does not collude with any of the DOs, because this would allow a much easier attack following the ideas of Mitchell et al. [139] we described in Chapter 4. Furthermore, we assume the LU can guess the domain of the sensitive databases depending on its knowledge about the DOs.

Existing attacks on PPRL also make similar assumptions about the knowledge of an adversary [41, 119, 121, 147, 216]. We assume that the adversary has access to an encoded database $\mathbf{D}^e = (\mathbf{E}, \mathbf{M})$ of $n_e = |\mathbf{D}^e|$ records where $\mathbf{E}$ is the list of encoded values and $\mathbf{M}$ is the list of mircodata for each record. $\mathbf{D}^e$ contains QID attribute values encoded using a PPRL technique such as BF, TMH, or 2SH discussed above. We also assume that the adversary has access to a plaintext database $\mathbf{D}^p = (\mathbf{Q})$ of $n_p = |\mathbf{D}^p|$ records, that is from the same domain as $\mathbf{D}^e$, where $\mathbf{Q}$ is the list of QID values for each record. As most other PPRL attacks do, we assume the adversary does know or can guess the types of QIDs that were encoded in $\mathbf{E}$ [180].

8.3.2 Generating Similarity Graphs

In the first step of the attack we generate two similarity graphs using the two lists $\mathbf{E}$ and $\mathbf{Q}$ from the databases $\mathbf{D}^e$ and $\mathbf{D}^p$ respectively. As can be seen from Figure 8.1, the main idea behind our attack is that given the two databases are from the same domain, then the two graphs will contain similar neighbourhoods for nodes that represent the same value (plaintext or encoded) across the two databases. In Algorithm 8.1 we illustrate the graph generation process using the plaintext list $\mathbf{Q}$. The encoded graph generation follows the same steps as the plaintext graph generation.

Consider an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges connecting those nodes. In a PPRL context, V corresponds to the set of unique values in a database and E to the similarities between those values. For a given pair of nodes $v_1 \in V$ and $v_2 \in V$, its edge attribute is $e_{1,2} = sim(v_1, v_2)$, where $sim()$ is the function used to calculate the similarity between two nodes (records). This can, for example, be the Dice or Jaccard similarity depending on the encoding technique used [158, 177, 186].

In Algorithm 8.1 we first initialise the undirected graph G^p with two empty sets of nodes and edges in line 1. Since calculating the similarity between every pair of records in $\mathbf{Q}$ can be time consuming, in line 2 we employ a suitable blocking mechanism such as locality sensitive hashing (LSH) to block the similar records together. In lines 3 to 6 we then loop over each block and each pair of records in these blocks, and add those records to the graph G^p as nodes. The function $addNode()$ in lines 13 to 19 adds the nodes to the corresponding graph. In the plaintext list $\mathbf{Q}$, a node

Algorithm 8.1: Generate plaintext similarity graph

Input : $\mathbf{Q}$: List of records with QID values from $\mathbf{D}^p$
s_m: Minimum similarity threshold
Output: G^p: Plaintext similarity graph

1 $G^p \leftarrow (V^p, E^p)$; $V^p \leftarrow \emptyset$; $E^p \leftarrow \emptyset$; // Initialise the plaintext similarity graph
2 $\mathbf{B}^p \leftarrow getSimilarRecordBlocks(\mathbf{Q})$ // Block similar plaintext records together
3 **for** $\mathbf{b}_a \in G^p$ **do**: // Loop over each block
4 **for** $(r_i, r_j) \in \mathbf{b}_a$ **do**: // Loop over each record pair in the block
5 $G^p, \mathbf{q}_i \leftarrow addNode(G^p, r_i)$ // Add the record r_i to G^p
6 $G^p, \mathbf{q}_j \leftarrow addNode(G^p, r_j)$ // Add the record r_j to G^p
7 $s_{i,j} \leftarrow sim(r_i, r_j)$ // Calculate the similarity between the two records r_i and r_j
8 **if** $s_{i,j} \geq s_m$ **then**: // If the similarity $s_{i,j}$ is at least s_m
9 **if** $(\mathbf{q}_i, \mathbf{q}_j) \notin G^p.E^p$ **then**: // If $(\mathbf{q}_i, \mathbf{q}_j)$ is not an edge in G^p
10 $G^p.E^p \leftarrow G^p.E^p \cup \{(\mathbf{q}_i, \mathbf{q}_j)\}$ // Add an edge between $\mathbf{q}_i$ and $\mathbf{q}_j$ to G^p
11 $G^p.E^p[\mathbf{q}_i, \mathbf{q}_j] \leftarrow s_{i,j}$ // Assign record pair similarity to the edge
12 **return** G^p
13 **Function** $addNode(G, r)$: // Add a new node or increase existing node frequency
14 $\mathbf{q} \leftarrow getQgramTuple(r)$ // Get the unique set of q-grams $\mathbf{q}$ of QID values
15 **if** $\mathbf{q} \notin G.V$ **then**: // If $\mathbf{q}$ is not a node in G
16 $G.V \leftarrow G.V \cup \{\mathbf{q}\}$ // Add a new node to G
17 $G.V[\mathbf{q}] \leftarrow 0$ // Initialise node ($\mathbf{q}$) frequency
18 $G.V[\mathbf{q}] \leftarrow G.V[\mathbf{q}] + 1$ // Increment node ($\mathbf{q}$) frequency by 1
19 **return** G, $\mathbf{q}$ // Return the graph and the unique set of q-grams

value is the unique set of q-grams extracted from the corresponding QID value(s), while in the encoded database $\mathbf{E}$ a node value is the unique set of random integers (2SH) or the unique bit vector (BF or TMH) representing an encoded QID value/s. In line 14 of the function $addNode()$ we get the unique set of q-gram tuple for the corresponding record using the function $getQgramTuple()$. However, there can be multiple records in $\mathbf{Q}$ that have the same q-grams set, and multiple records in $\mathbf{E}$ that have the same encoding. In such scenarios we use a single node to represent all records with that q-gram set or encoding, and we set the node frequency as the number of records with that q-gram set or encoding as shown in lines 15 to 18. For example, if there are 10 records with the same first and last name a single node will represent all these records (since they will all have the same q-gram set) where we set the node frequency to 10.

To limit the number of edges between nodes in the graphs we generate, we use a minimum similarity threshold s_m, with $0 \leq s_m \leq 1$ (assuming $0 \leq sim(v_1, v_2) \leq 1$). As shown in lines 8 to 11 we only consider similarities between nodes that are at least s_m and those similarities are added to the set of edges in G^p. Higher values of s_m mean the generated graphs will contain less edges.

We denote the two similarity graphs generated from the two lists $\mathbf{E}$ and $\mathbf{Q}$ as $G^e = (V^e, E^e)$ and $G^p = (V^p, E^p)$. The encoded similarity graph will contain $|V^e| \leq n_e$ nodes and $|E^e| \leq n_e(n_e - 1)/2$ edges, whereas the plaintext similarity graph will contain $|V^p| \leq n_p$ nodes and $|E^p| \leq n_p(n_p - 1)/2$ edges.

Table 8.2: The node, edge, and neighbourhood features we generate for each node in graphs G^e and G^p where s_m is the minimum similarity threshold considered in the graph generation step (as discussed in Section 8.3.2) and c_m is the minimum connected component size considered in the feature generation step (as discussed in Section 8.3.3).

NodeFreq	The number of records in **Q** or **E**, respectively, that contain the value represented by a node $v \in V$.
NodeLen	Length of q-gram set (in plaintext graph), integer encoding set for 2SH, or number of 1-bits (Hamming weight) for BF and TMH.
$\text{NodeDegr}_{s_m,c_m}$	Number of edges connected to a node $v \in V$ for given minimum similarity threshold s_m and minimum connected component size c_m.
EdgeMax	Maximum similarity of the edges connected to a node $v \in V$.
$\text{EdgeMin}_{s_m,c_m}$	Minimum similarity of the edges connected to a node $v \in V$, for given s_m and c_m.
$\text{EdgeAvr}_{s_m,c_m}$	Arithmetic average of similarities of the edges connected to a node $v \in V$, for given s_m and c_m.
$\text{EdgeStdDev}_{s_m,c_m}$	Standard deviation of similarities of the edges connected to a node $v \in V$, for given s_m and c_m.
$\text{EgonetDegr}_{s_m,c_m}$	Degree of the induced subgraph for a node $v \in V$ which includes v and all of its direct neighbours, for given s_m and c_m.
$\text{EgonetDens}_{s_m,c_m}$	Density of the induced subgraph for a node $v \in V$ which includes v and all of its direct neighbours, for given s_m and c_m.
$\text{BetwCentr}_{s_m,c_m}$	Betweenness centrality, the ratio of the number of shortest paths that traverse through a node $v \in V$ to all shortest paths, for given s_m and c_m.
$\text{DegrCentr}_{s_m,c_m}$	Degree centrality, the ratio of the nodes connected to a node $v \in V$ with regard to all nodes in the connected component, for given s_m and c_m.
$\text{OneHopHisto}_{s_m,c_m}$	Degree distribution of the one-hop neighbour nodes of a node $v \in V$, for given s_m and c_m.
$\text{TwoHopHisto}_{s_m,c_m}$	Degree distribution of the two-hop neighbour nodes of a node $v \in V$, for given s_m and c_m.

8.3.3 Node Feature Generation

The second step of the attack generates features for each node in V^e and in V^p. These features, as listed in Table 8.2, represent the characteristics of a given node based on its value and the edges connecting it to neighbouring nodes. We only consider nodes that are part of a connected component with a minimum size c_m (based on a minimum similarity threshold s_m on the edges in a graph, as discussed above), because otherwise the nodes would not contain enough information to allow the generation of features which will help the attack to distinguish between nodes. For example, all nodes with no or only one edge connecting them to a single neighbouring node will likely have the same feature values making them indistinguishable from each other in the graph matching step.

For each node in the two graphs we generate a feature vector, where our attack is based on identifying highly similar feature vectors to match pairs of plaintext and encoded values. The generated feature vectors will differ for different minimum similarity thresholds s_m and minimum connected component sizes c_m, because the neighbourhoods of nodes will be different with different number of edges connect-

ing them. We discuss what values for s_m and c_m provided the best results in our experiments in Section 8.6. We generate three types of features, as shown in Table 8.2 and described next.

- **Node based Features**: Each node in the similarity graphs G^e and G^p has two attributes, its frequency and length. Node frequency is how many records in **E** or **Q**, respectively, contain a certain q-gram set or encoded value, as we discussed above. As a node's length, for the plaintext graph G^p we use the number of q-grams in a node, while for a node in G^e we use the Hamming weight (number of 1-bits) of the bit vectors for BF and TMH encoding, and the size of the integer set generated for 2SH.

- **Edge based Features**: For a graph generated based on given values of s_m and c_m, each node that is part of a connected component of size at least c_m will have one or more neighbouring nodes it is connected to. We calculate edge features for each such node v_i based on the set of its neighbouring nodes as $N(v_i) = \{v_j | (v_i, v_j) \in E\}$, and the set of edges connecting each neighbour $v_j \in N(v_i)$ to v_i as $E(v_i) = \{e_{i,j} | v_j \in N(v_i)\}$. As listed in Table 8.2, these edge features are the degree of the node v_i, $d(v_i) = |N(v_i)|$, and the minimum, $min(E(v_i))$, maximum, $max(E(v_i))$, average, $avg(E(v_i)) = (\sum_{j=1}^{|E(v_i)|} e_{i,j})/|E(v_i)|$, and standard deviation, $std(E(v_i))$, of neighbouring edge values.

- **Structural Features**: Under structural features we consider all nodes that are part of a connected component, not just the direct neighbours of a given node v_i. We calculate egonet degree [3] and egonet density using the egonet of a node v_i. For a given node $v_i \in V$, the egonet of that node is the subgraph G' of G where $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$ which includes the node v_i, all its direct neighbours, and all the edges between them. We also calculate centrality metrics such as betweenness centrality and degree centrality [3]. These centrality metrics measure the importance of a node in the graph based on the connectivity of that node within the graph. We also consider the degree distribution of a node's one-hop and two-hop neighbourhoods. We calculate neighbourhood node degree histograms based on a logarithmic scale for each node [92] and use them as features.

At the end of the feature generation process each node in V^e and in V^p will have a vector $\mathbf{f}$ of features assigned to it where $m = |\mathbf{f}|$ is the number of features considered. We denote the resulting two feature matrices for databases **E** and **Q** as $\mathbf{F}^e$ and $\mathbf{F}^p$, respectively.

8.3.4 Node Matching

After generating the feature matrices $\mathbf{F}^e$ and $\mathbf{F}^p$, we next match nodes between the two graphs G^e and G^p to identify encoded values in **E** that correspond to plaintext values in **Q**. We now describe the steps taken to perform node matching in an

effective and efficient manner. Given we are processing feature vectors in both $\mathbf{F}^e$ and $\mathbf{F}^p$ in the same way, to improve readability we refer to both as $\mathbf{F}$.

Many graph matching and alignment algorithms have been developed in research domains ranging from bioinformatics to multimedia information retrieval [72]. Such algorithms can be based on graph kernels [116], node and edge embeddings [83], or hashing [91]. In our approach, as we discuss below, we use a Cosine LSH based approach because it allows us to explicitly set a trade-off between the performance of the attack (as the number of feature vectors to be compared) versus the probability of comparing similar nodes [91].

8.3.4.1 Reducing Comparison Space using Locality Sensitive Hashing

With n_e nodes in the encoded graph G^e and n_p nodes in the plaintext graph G^p, there can be a maximum of $n_e \cdot n_p$ pairs of feature vectors for which we would need to calculate their similarities. This is not scalable for large databases. We therefore use LSH to reduce the comparison space. Since the node feature vectors, $\mathbf{f}$, are real-valued we use Cosine similarity based LSH [8]. We first randomly generate d column vectors (hyperplanes) $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_d\} \in \mathbb{R}^m$ that follow a uniform stable distribution such as a Gaussian distribution. We then use these column vectors as the LSH family of hash functions $\mathcal{H}$ to map the nodes into blocks using the feature vector $\mathbf{f} \in \mathbf{F}$ of each node.

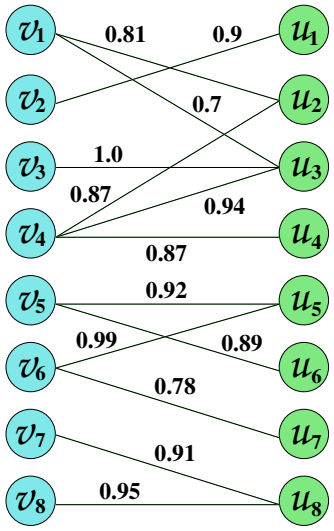
The mapping of node v_i, $h_d(v_i)$, where $h() \in \mathcal{H}$, is based on the sign of the dot product $\mathbf{f}_{v_i} \cdot \mathbf{c}_j$:

$$h_d(v_i) = \begin{cases} 0, & \text{if } \mathbf{f}_{v_i} \cdot \mathbf{c}_j < 0 \\ 1, & \text{otherwise.} \end{cases} \tag{8.1}$$

where $\mathbf{f}_{v_i}$ is the feature vector of the node v_i and $\mathbf{c}_j$ is a random column vector $1 \leq j \leq d$. Given there are d such column vectors, each feature vector $\mathbf{f} \in \mathbf{F}$ of a node is therefore mapped to a binary vector space $\mathcal{H}(v_i) = \{0,1\}^d$ resulting in a bit vector of length d assigned to each node. We next use Hamming LSH on these mapped bit vectors to group the nodes in the two graphs G^e and G^p into similar blocks. Using this method the attack reduces the quadratic comparison space by removing likely non-matching nodes and grouping potential matching nodes into the same blocks.

8.3.4.2 Bi-partite Graph Matching

After the LSH based blocking step, the comparison of nodes across the two graphs will become a bi-partite graph matching problem [3, 25]. We compare pairs of encoded and plaintext nodes from similar blocks using a defined similarity metric, and then select pairs with a similarity higher than a user defined threshold b_m. Since node pairs are compared using their respective feature vectors, we use the Cosine similarity to calculate their similarity. This results in a bi-partite graph $G^b = (V^b, U^b, E^b)$ where $V^b \subseteq V^e$ and $U^b \subseteq V^p$. It is not necessary to have $|V^b| = |U^b|$ and there can be multiple edges connecting one encoded node to multiple plaintext nodes (and



Edges	Cosine similarity	Similarity confidence	Degree confidence
(v_1, u_2)	0.81	1.03	0.33
(v_1, u_3)	0.7	0.76	0.25
(v_2, u_1)	0.9	1.5	1.0
(v_3, u_3)	1.0	1.22	0.33
(v_4, u_2)	0.87	0.99	0.25
(v_4, u_3)	0.94	1.09	0.2
(v_4, u_4)	0.87	0.96	0.33
(v_5, u_5)	0.92	0.98	0.33
(v_5, u_6)	0.89	0.97	0.5
(v_6, u_5)	0.99	1.17	0.33
(v_6, u_7)	0.78	0.79	0.5
(v_7, u_8)	0.91	0.96	0.5
(v_8, u_8)	0.95	1.04	0.5

Figure 8.2: An example bi-partite graph matching problem, where encoded nodes v_i are connected to plaintext nodes u_i. For each edge the calculated Cosine similarity, similarity confidence, and degree confidence are shown, as discussed in Section 8.3.4.

vise-versa) in G^b, as shown in Figure 8.2. Note that from here onward we refer to a node in V^b as v and to a node in U^b as u to improve readability.

We select a set of edges, $E^s = \{(v,u), v \in V^b, u \in U^b\}$, where $E^s \subseteq E^b$, between encoded and plaintext nodes in such a way that for every pair of edges $e_1, e_2 \in E^s$, e_1 and e_2 do not share common nodes. Formally, $\exists (v,u) \in E^s \iff \forall (v',u') \in E^s : (v,u) \neq (v',u'); (v' \neq v) \wedge (u' \neq u)$. Figure 8.2 shows an example bi-partite matching problem of two sets of nodes from G^e and G^p. We define three different edge weights: (1) Cosine similarity, (2) similarity confidence, and (3) degree confidence that can be used when matching nodes in G^b.

The Cosine similarity, $cs(v,u)$, of a given edge is calculated as [91]:

$$cs(v,u) = \frac{\mathbf{f}_v \cdot \mathbf{f}_u}{\|\mathbf{f}_v\| \cdot \|\mathbf{f}_u\|}, \tag{8.2}$$

where $\mathbf{f}_v$ and $\mathbf{f}_u$ are the normalised feature vectors calculated for the two nodes v and u and $\|-\|$ calculates the Euclidean norm of a vector. The similarity confidence, $cs(v,u)$, measures the reliability of an edge value with respect to all the other values of the edges connected to the nodes v and u. The similarity confidence, $sc(v,u)$, of a given edge is calculated as:

$$sc(v,u) = \frac{cs(v,u) \cdot (\alpha + \beta - 2)}{\sum_{i=1}^{\alpha-1} cs(v,u_i) + \sum_{j=1}^{\beta-1} cs(v_j,u)}, \tag{8.3}$$

where $cs(v,u)$ is the Cosine similarity of the node pair (v,u), α is the degree of the encoded node v, β is the degree of the plaintext node u, $\alpha - 1$ is the number of other

edges connected to v, and $\beta - 1$ is the number of other edges connected to u. The degree confidence, $dc(v, u)$ measures the reliability of an edge (v, u) with respect to the number of other edges connected to the nodes v and u. The degree confidence of a given edge is calculated as:

$$dc(v, u) = \frac{1}{\alpha + \beta - 1}, \tag{8.4}$$

where $\alpha + \beta - 1$ is the total number of edges connected to both v and u. In Figure 8.2 we show the calculated values for these three edge weights in the given bi-partite graph. Note that since the numerical values calculated by these three weight types are not in the same range, we normalise them using min-max normalisation[1] prior to matching.

Using the above three weights defined for each edge in G^b we then aim to match encoded nodes with plaintext nodes. The three types of weights can be combined in a weighted way, using w_{cs}, w_{sc}, and w_{dc}, where $w_{cs} + w_{sc} + w_{dc} = 1.0$, based on their importance in being able to identify the correct plaintext node for an encoded node. We discuss the suitability of these three weight types to correctly match nodes across the two graphs in Section 8.6.3.

The actual matching of nodes can be performed using different methods. Franke et al. [74] proposed three methods of performing one-to-one matching of nodes between a bi-partite graph.

- **Symmetric Highest Match (SHM)**: The SHM method finds, for each node in both encoded and plaintext graphs, the edge that has the highest weight. Here the edge weight can be a value calculated using one of the above three weight types, or a weighted average of the three weight types. For instance in Figure 8.2 the encoded node v_6 and the plaintext node u_5 have three edges (v_6, u_5), (v_6, u_7), and (v_5, u_5) connected to them. In Figure 8.3, when we calculate the weighted edge values we get $(v_6, u_5) = 0.68$, $(v_6, u_7) = 0.22$, and $(v_5, u_5) = 0.49$. As can be seen, since the highest match for both v_6 and u_5 is the edge connecting these two nodes it is selected, while all the other edges connected to those two nodes are discarded.

- **Stable Marriage Match (SMM)**: The SMM method solves the node matching problem by finding stable matches in the bi-partite graph G^b. A matched edge is defined as *stable* if there are no other edges connecting both nodes that have higher weight than the matched edge. We use the Gale–Shapley algorithm [78] to solve this problem. However, compared to the original algorithm, our specific case of matching nodes from the encoded graph to the plaintext graph requires two modifications. The original algorithm assumes that each node from one side of the graph G^b has a complete preference list including every node from the other side. As can be seen from Figure 8.2 this is not the case for our problem. The original algorithm also assumes equal sized sets of nodes in

[1] https://en.wikipedia.org/wiki/Feature_scaling

Algorithm 8.2: Bi-partite graph matching with symmetric highest match

Input : G^b: Encoded/plaintext bi-partite graph
w_{cs}, w_{sc}, w_{dc}: Weight values for weighted edge calculation
Output: E^s: Set of selected edges

1 $L = [\,]$; $E^s = \{\}$; // Initialise the list L of weighted edges and the set E^s of final selected weighted edges
2 **for** $v_i, u_j \in G^b.V^b, G^b.U^b$ **do**: // Loop over each encoded/plaintext node pair (or edge)
3 $s_{cs} \leftarrow G^b.E^b[v_i, u_j]$ // Get the Cosine similarity
4 $s_{sc} \leftarrow calcSimConf(v_i, u_j)$ // Calculate similarity confidence
5 $s_{dc} \leftarrow calcDegreeConf(v_i, u_j)$ // Calculate degree confidence
6 $s_{av} \leftarrow s_{cs} \cdot w_{cs} + s_{sc} \cdot w_{sc} + s_{dc} \cdot w_{dc}$ // Calculate weighted edge value
7 $L.append((v_i, u_j, s_{av}))$ // Add weighted edge to L
8 $A^v = \{\}$; $A^u = \{\}$; // Initialise the sets of traversed encoded and plaintext nodes
9 $L.sort_largest_edge_value_first()$ // Sort the list L according to the edge values
10 **for** $v_i, u_j, s_{av} \in L$ **do**: // Loop over each edge
11 **if** $v_i \notin A^v \wedge u_j \notin A^u$ **then**: // If both edges are not already traversed
12 $E^s.add((v_i, u_j, s_{av}))$: // Add edge to the selected list of edges
13 $A^v.add(v_i)$: // Add encoded edge to A^v
14 $A^u.add(u_j)$: // Add plaintext edge to A^u
15 **return** E^s

both side of the graph G^b. This might not be always true in our case because we use a minimum connected component size threshold c_m and LSH based blocking to group similar nodes. We have modified the original algorithm to allow both of these changes and identify the stable matches across the bi-partite graph G^b.

- **Maximum Weight Match (MWM)**: The MWM corresponds to the assignment problem in a bi-partite graph where the set of edges E^s is selected in such a way that the sum of the weights of the selected edges are maximised. This assignment problem can be solved using the Kuhn-Munkres algorithm (also known as the Hungarian algorithm) [120] in polynomial time.

In Algorithm 8.2 we show the bi-partite graph matching process using the SHM method. We first initialise a list L for keeping the weighted edges in the graph G^b. From line 2 to 6, for each edge, we then calculate the different edge weights Cosine similarity, (s_{cs}), similarity confidence, (s_{sc}), and degree confidence, (s_{dc}), and then the weighted edge value s_{av} as we discussed above. In line 7 we append s_{av} along with the corresponding nodes to the list L. In line 8 to 14 we finally apply the SHM method to identify the node pairs with highest weights. As can be seen, we do not select node pairs (v_i, u_j) where the edge between them is the highest for only one of the nodes. In line 15 we finally return the selected set of edges E^s using the SHM node matching method.

In Figure 8.3 we show example graph matching results obtained using the above discussed three matching techniques with weight values $w_{cs} = 0.5$, $w_{sc} = 0.3$, and

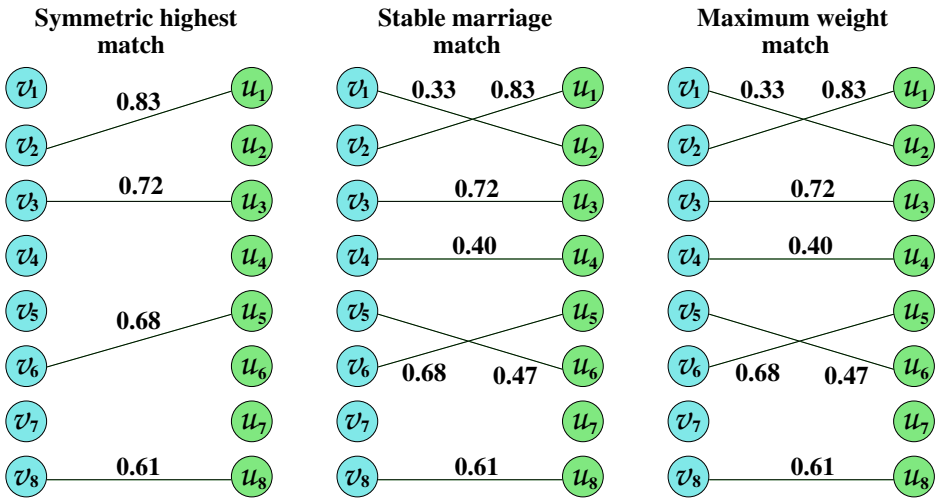


Figure 8.3: An example solution of the bi-partite graph matching problem from Figure 8.2 using the weights $w_{cs} = 0.5$, $w_{sc} = 0.3$, and $w_{dc} = 0.2$ for Cosine similarity, similarity confidence, and degree confidence, respectively.

$w_{dc} = 0.2$ for each edge weighting technique. At the end of this matching step, no node in the resulting set of edges, E^s, will be included in more than one edge.

8.3.5 Reidentification of Plaintext Values

Once enocoded and plaintext nodes are matched in the bi-partite graph G^b, the final step of our attack is to perform plaintext value reidentification. Since the selected set of edges E^s from the previous step only consists of one-to-one matches, we can directly align the plaintext value associated with a node from G^p to its matched encoded node in G^e using the edges E^s in G^b.

First the attack sorts all the edges in $(v, u) \in E^s$ according to their edge weights in descending order. Then we loop over each edge and retrieve the corresponding encoded and plaintext values from the lists **E** and **Q**, respectively. We continue this process until the similarity between pairs of nodes is below a similarity threshold s_t, where s_t is a user defined value. With smaller s_t we will potentially match and reidentify incorrect values as their similarities become lower. In this assignment process there can be 1-to-*many*, *many*-to-1, or *many*-to-*many* matches of records as a node from either graph, G^p or G^e, can represent more than one record (as discussed in Section 8.3.2), where however all these records have the same plaintext or encoded value. We discussed different types of matches in more detail in Section 3.7.1.

8.4 Adjusting Similarities

For our attack method to work accurately, it is important to have similar neighbourhood structures for nodes representing the same value across the two graphs. Assuming that the adversary can estimate the parameter values used in the encoding

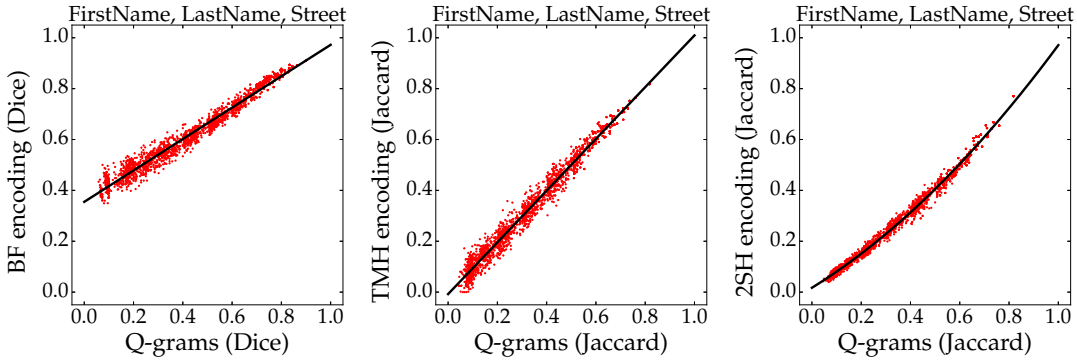


Figure 8.4: The relationship between encoded and plaintext similarities using the encoding techniques BF (left), TMH (centre), and 2SH (right). The regression models (black lines) are build using a NCVR-20k database (as discussed in Section 8.6).

of values (even if not all parameter values are set correctly), she can perform a regression analysis using the plaintext database to determine the relationship between the encoded and plaintext similarity distributions. Some encoding parameters, such as BF length l, can be learned by analysing the encoded values themselves. Other parameters, such as the number of hash functions used, k, can be approximated by comparing the Hamming weight distribution of BFs to the distribution of the number of q-grams in plaintext values. By referring to previous publications and/or real-world applications, where encoding techniques are used for PPRL, the likely used parameter settings can also be obtained. It is important to note that by obtaining these parameters an adversary cannot simply reconstruct the exact encodings (BF, THM, or 2SH) because the hash functions and secret key used will not be known to the adversary.

In Figure 8.4 we show three example regression models build based on the three encoding techniques. As can be seen, the relationships between the encoded and plaintext similarities are approximately linear with BF and TMH encoding, and slightly non-linear with 2SH encoding. However, BF similarities are significantly higher than the corresponding plaintext q-gram set similarities due to collisions [158]. Using such regression models, the adversary can adjust plaintext similarities such that they align with encoded similarities when generating the plaintext graph G^p. Using this additional step during the similarity graph generation of our attack can significantly increase the accuracy of the attack because when encoded and plaintext similarities are aligned the overall structure of the two generated graphs, G^e and G^p, will be more similar. This leads to the corresponding nodes in the encoded and plaintext graphs to have highly similar neighbourhoods, and therefore highly similar feature vectors. In Section 8.6.3 we discuss the accuracy improvements we were able to gain using this technique.

8.5 Complexity Analysis

We evaluate the worst-case computational complexity of our attack by analysing the number of computational steps required for the each step of the attack discussed in Sections 8.3.2 to 8.3.5. The complexity of the graph generation step of the attack (Algorithm 8.1) is as follows. The complexity of the record blocking in line 2 of Algorithm 8.1 depends on the blocking technique used. For instance sorted neighbourhood blocking has $O(n)$ complexity and LSH based blocking has $O(log\ n)$ where n is the number of records in a database. The main loop starting from line 3 has $O(n_p(n_p-1)/2)$ or $O(n_e(n_e-1)/2)$ for the encoded similarity graph generation) worst-case complexity assuming all possible record pairs are inside the blocks. But in reality this will not be the case because blocking will reduce the number of pair-wise record comparisons significantly [38].

Node feature generation step has $O(|V^p| + |V^e|)$ complexity where $|V^p|$ and $|V^e|$ are the numbers of nodes in graphs G^p and G^e, respectively. Note that each of the features we calculate for each node has its own complexity. Feature like the 'NodeFreq' has $O(1)$ complexity whereas the 'BetwCentr' algorithm we use [30] has $O(|V^p| \cdot |E^p| + |V^p|^2\ log\ |V^p|)$ complexity (for a plaintext node).

The complexity of the node matching step of the attack is as follows. Calculating the Cosine similarity between nodes has $O(|V^p||V^e|)$ complexity assuming all possible pairs of nodes from the graphs G^p and G^e are compared. The bi-partite graph matching step will have different complexities depending on the algorithm used. The SHM algorithm has $O(2m)$ complexity where $m = min(|V^b|, |U^b|)$ and the function $min()$ selects the minimum of the two values. The Gale-Shapley algorithm (SMM) has $O(m^2)$ complexity [98] whereas the Kuhn-Munkres algorithm (MWM) has $O(m^3)$ complexity [70].

The plaintext reidentification step of the attack has $O(m)$ complexity under the assumption that all edges connecting pairs of nodes in G^b have at least the minimum similarity s_t.

8.6 Experimental Evaluation

We conducted an experimental evaluation of our proposed graph matching attack on PPRL encoding techniques using a real-world database. We measured the accuracy of the attack under different parameter settings to investigate the feasibility of our attack in real-world linkage situations.

8.6.1 Databases and Experimental Setup

As we discussed in Chapters 6 and 7, we used a snapshot of North Carolina Voter Registration (NCVR) database[2] to create twenty sensitive ($\mathbf{D}^p$) and plaintext ($\mathbf{D}^s$) database pairs. As mentioned, we created five sets of database pairs containing 20,000

[2] http://dl.ncsbe.gov/data/

(20k), 100,000 (100k), 200,000 (200k), 500,000 (500k), and 1,000,000 (1M) records and in each set we had four database pairs which had 100%, 75%, 50%, and 0% matching records between them. We used different combinations of the attributes *FirstName*, *LastName*, *StreetAddress*, and *City* to generate different instances of encoded databases using the three encoding techniques BF, TMH, and 2SH. These attributes are commonly used in PPRL to link records across different databases [209].

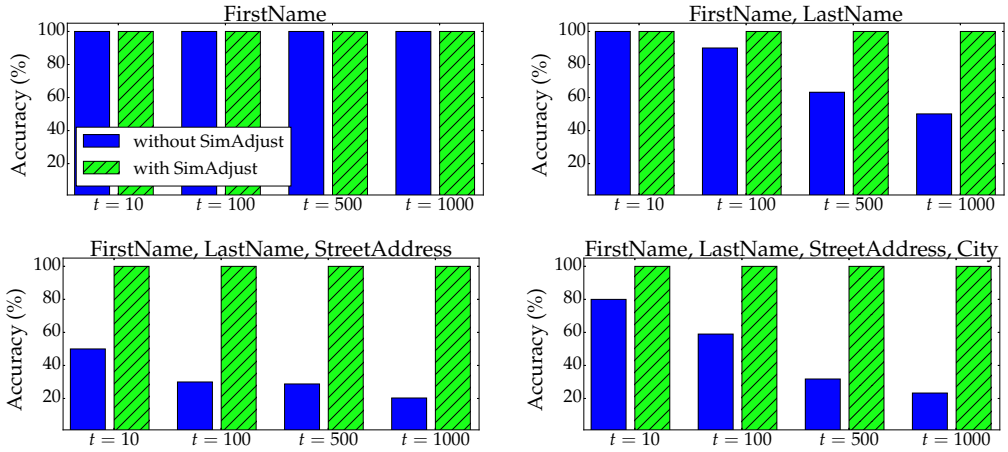
Following earlier work [27, 64, 177], for BF encoding we used the parameter settings: $l_q = 2$, $l = 1,000$, and $k = [10, 15, 30, opt]$. BFs were encoded using both the CLK and RBF approaches [61, 178]. Following [186], for TMH encoding we used $c = [4, 8]$ look-up tables per min-hash signature, where in each look-up table bit keys of length $k = [4, 8]$ were used to refer to 32 and 64 bit random bit strings, respectively. Using the obtained min-hash signatures, bit arrays of length $l = 1,000$ were generated for each encoded value [186]. For 2SH encoding, we used the parameter settings: $l_q = 2$, $l = 1,000$, and $k = [10, 15, 30]$ [158]. The accuracy results discussed below for each database were averaged over all these parameter settings.

We set $s_m = [0.2, 0.3, 0.5]$ for the minimum similarity threshold and $c_m = [5, 10, 50]$ for the minimum connected component size threshold for the graph generation. Furthermore, we set $b_m = [0.8, 0.9]$ as the minimum similarity threshold when generating the bi-partite graph as discussed in Section 8.3.4.2. For the plaintext reidentification we set $s_t = [0.6, 0.7, 0.8]$ minimum similarity threshold for selecting matching nodes. For the alignment of similarities between encoded and plaintext databases we used polynomial regression [142] and built regression models using a sample size of 3,000 to 5,000 edge (similarity) pairs from the plaintext graph. These parameter values provided accurate results in a series of set-up experiments.

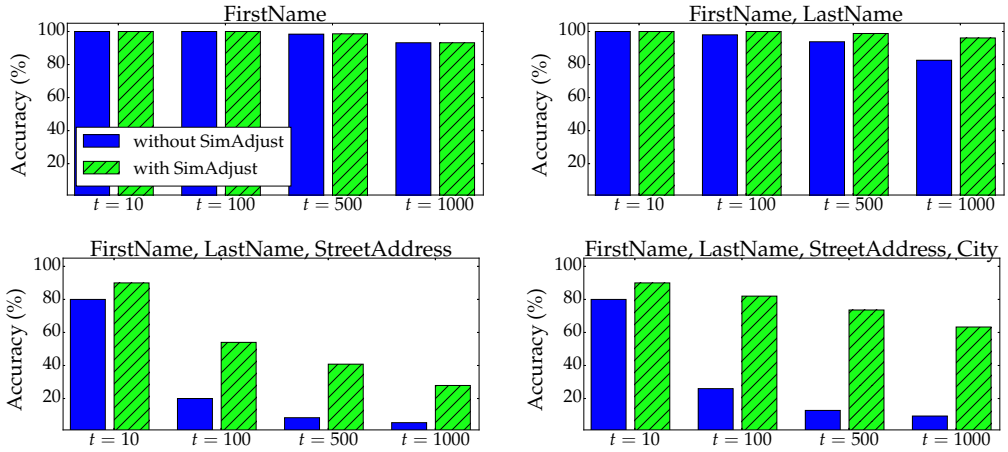
8.6.2 Evaluation Criteria

We first evaluated the similarity adjustment step (as discussed in Section 8.4) by comparing the attribute reidentification results for both without and with similarity adjustments. For this we analysed and calculated the percentage of correct top one-to-one t reidentifications (when sorted according to the edge weight between encoded and plaintext nodes) where $t = [10, 100, 500, 1000]$. We also compared the reidentification results for the three different bi-partite graph matching techniques, SHM, SMM, and MWM, and the three different bi-partite graph edge weighting approaches, $cs()$, $sc()$, and $dc()$ to evaluate their performance.

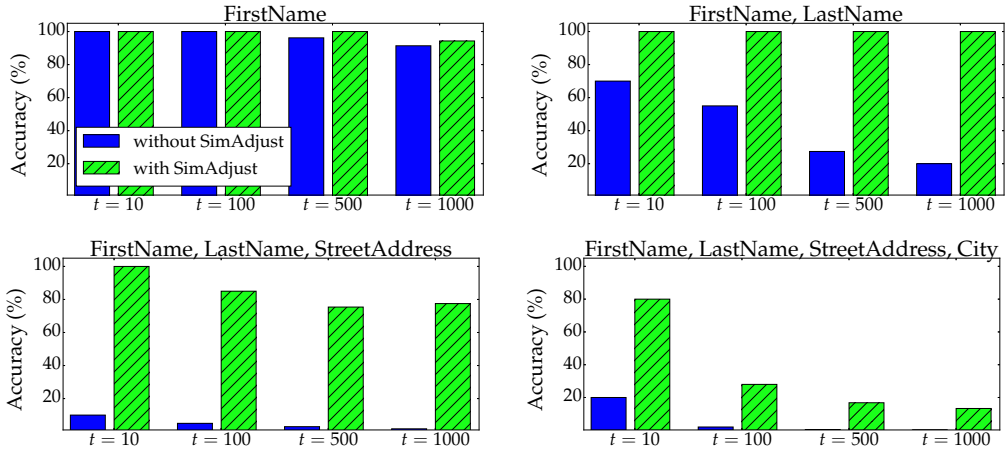
Like the previous two attack methods discussed in Chapters 6 and 7, we evaluated the accuracy of the reidentified plaintext QID values for encodings and reidentified entities using the number of 1-to-1, 1-to-*many*, *many*-to-1, and *many*-to-*many* reidentifications as we discussed in Chapter 3. We also calculated the probability of suspicion (PS) values for each record in the encoded database $\mathbf{D}^e$ as discussed in Section 3.7.1.



(i) Accuracy results of BF encoding.



(ii) Accuracy results of TMH encoding.



(iii) Accuracy results of 2SH encoding.

Figure 8.5: Accuracy results for reidentified plaintext QID values on the NCVR-20k database pairs, both without and with similarity adjustments as discussed in Section 8.4. The results are presented for different attribute combinations.

Table 8.3: Percentages of accurate attribute reidentifications using different bi-partite graph matching and edge weighting methods, as discussed in Section 8.3.4, for the NCVR-20k database pairs with respect to all 20,000 potential matches. Percentages are shown without / with similarity adjustments applied. In the *all-three* edge weighting method we used the weights $w_{cs} = 0.6$, $w_{sc} = 0.3$, and $w_{dc} = 0.1$ for Cosine similarity, similarity confidence, and degree confidence, respectively.

Attribute combination	Encoding technique	Graph matching method			Graph edge weighting method			
		SHM	SMM	MWM	$cs(v,u)$	$sc(v,u)$	$dc(v,u)$	*all-three*
FirstName	BF	76.5 / 85.4	76.9 / 85.4	77.0 / 85.4	76.8 / 85.4	77.0 / 85.4	70.3 / 84.8	76.9 / 85.4
	TMH	8.6 / 35.5	8.7 / 37.2	8.7 / 36.0	8.6 / 36.6	8.7 / 36.8	8.3 / 25.9	8.7 / 37.2
	2SH	31.4 / 20.9	32.3 / 21.0	31.2 / 20.9	31.9 / 21.0	32.1 / 21.0	24.9 / 20.1	32.3 / 21.0
FirstName and *LastName*	BF	5.2 / 93.2	6.0 / 93.2	4.2 / 93.2	5.7 / 93.2	5.8 / 93.2	4.8 / 91.2	6.0 / 93.2
	TMH	7.1 / 15.6	7.1 / 15.9	7.0 / 15.2	7.1 / 15.8	7.1 / 15.7	5.9 / 11.7	7.1 / 15.9
	2SH	1.6 / 27.0	1.8 / 27.1	1.6 / 26.2	1.7 / 27.1	1.8 / 27.1	1.4 / 22.6	1.8 / 27.1

8.6.3 Results and Discussion

In Figure 8.5 we show accuracy results for the NCVR-20k database pairs with four different attribute combinations, three encoding techniques, and both without and with similarity adjustments (as discussed in Section 8.4). As can be seen, the attack was more successful with similarity adjustment compared to without similarity adjustment. With one and two attributes the attack consistently achieved high reidentification accuracy. However, the results both without and with similarity adjustments drop when more attributes are encoded because with more attributes the neighbourhoods of dissimilar nodes (which corresponds to non-matching values) across the two graphs G^e and G^p become less distinct as they get connected by more edges. This can be seen in the results for all three encoding techniques.

Out of the three encoding techniques, on average the attack on TMH encoding achieved the best results without similarity adjustments. This is because the similarities between TMH encodings are highly similar to the corresponding plaintext q-gram set similarities (as can be seen in the centre plot of Figure 8.4). This results in encoded and plaintext graphs, G^e and G^p, with highly similar node neighbourhoods. However, the attack on BF encoding provided the best reidentification accuracy when similarity adjustment is applied because of the linear relationship between encoded and plaintext similarities. When attacking 2SH encoding, the accuracy of reidentifications both with and without similarity adjustment drops because of the similarity differences as well as the non-linear relationship between encoded and plaintext similarities (as shown in Figure 8.4). Overall, our attack achieved better reidentification accuracy with similarity adjustment compared to without, and with one and two attribute combinations compared to three and four.

In Table 8.3 we show the percentages of accurate one-to-one attribute reidentifications (averaged over all other parameter settings) for the three different bi-partite graph matching techniques, SHM, SMM, and MWM, and the three different bi-partite graph edge weighting approaches, $cs()$, $sc()$, and $dc()$. Overall, the SMM matching method performed slightly better compared to other two matching methods, where

Table 8.4: Attribute reidentification accuracy results for the NCVR-100k database pairs with different number of matching records. We present the results for all four attribute combinations and for database pairs with 100%, 75%, and 50% matching records between them.

Attribute combination	Encoding technique	100% / 75% / 50% 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FirstName	BF	12,963/335/627	8/0/0	0/0/0	0/0/0	22/941/458
	TMH	2,874/723/233	2/1/3	0/0/0	0/0/0	2,725/3,701/3,450
	2SH	5,024/1,056/458	4/3/2	0/0/0	0/0/0	1,395/3,455/3,585
FirstName and *LastName*	BF	41,390/2,018/1,344	26/1/1	0/0/0	0/0/0	878/20,171/38,855
	TMH	4,162/1,779/1,446	2/2/3	0/0/0	0/0/0	24,812/29,042/46,595
	2SH	20,498/7,547/2,383	17/6/2	0/0/0	0/0/0	19,611/40,323/44,115
FirstName, *LastName*, and *StreetAddress*	BF	46,884/330/253	0/0/0	0/0/0	0/0/0	13,517/32,077/61,786
	TMH	100/45/24	0/0/0	0/0/0	0/0/0	13,321/24,561/27,295
	2SH	346/72/32	0/0/0	0/0/0	0/0/0	15,574/21,638/22,095
FirstName, *LastName*, *StreetAddress*, and *City*	BF	50,852/697/166	0/0/0	0/0/0	0/0/0	10,678/52,300/183
	TMH	606/279/43	0/0/0	0/0/0	0/0/0	45,878/48,634/31,828
	2SH	396/116/41	0/0/0	0/0/0	0/0/0	14,209/21,082/23,027

SHM and MWM performed relatively similar. Furthermore, the experiments with the MWM matching consistently took considerably longer time compared to SHM and SMM. This is because of the high computational complexity of the Kuhn-Munkres algorithm as we discussed in Section 8.5.

From the three edge weighting approaches, $cs()$ and $sc()$ performed better than $dc()$ in all experiments, where the $sc()$ approach performed slightly better than $cs()$. However, in general the weighted use of all three approaches (*all-three*) provided better results compared to using a single approach only. Similar results were obtained for the other two attribute combinations.

In Tables 8.4 and 8.5 we show attribute and identity reidentification results of the attack, respectively, for all four attribute combinations and for database pairs with different matching records. As can be seen the attack was able to reidentify a considerably large amount of QID values (minimum 233 to maximum 41,390) and entities (minimum 98 to maximum 39,376) with only one and two attribute combinations. However, with three and four attribute combinations the reidentification results were comparatively low only for TMH and 2SH encodings. This is because with more attributes the graphs becomes dense making non-matching nodes to become more similar to each other as we also discussed above. However, with BF encoding the attack was able to reidentify a large number of values and entities (a maximum of 50,852) because the similarity adjustment step worked better with BF encoding compared to TMH and 2SH encodings. The reidentification accuracy drops significantly when there are less matching records between the encoded and plaintext databases. With 0% matching records between databases the attack was not able to identify a single QID value or entity correctly. This is due to the high sensitivity of the node features in the graphs. With non-matching records between databases the neighbourhoods of the nodes will be different. Even a small change in the neigh-

Table 8.5: Identity reidentification accuracy results for the NCVR-100k database pairs with different number of matching records. We present the results for all four attribute combinations and for database pairs with 100%, 75%, and 50% matching records between them.

Attribute combination	Encoding technique	100% / 75% / 50%				
		1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FirstName	BF	9,191/213/274	0/12/46	0/15/34	3,780/93/216	22/943/464
	TMH	1,951/408/98	0/28/23	0/30/10	925/250/85	2,725/3,712/3,472
	2SH	3,454/605/199	0/43/31	0/50/26	1,575/345/168	1,395/3,473/3,623
FirstName and *LastName*	BF	39,376/1,841/1,154	0/42/39	0/32/40	2,041/97/70	878/20,180/38,902
	TMH	3,906/1,559/1,120	0/38/57	0/35/63	258/140/136	24,812/29,056/46,671
	2SH	19,279/6,622/1,837	0/147/102	0/162/91	1,236/568/210	19,611/40,379/44,257
FirstName, *LastName*, and *StreetAddress*	BF	46,880/330/253	0/0/0	0/0/0	4/0/0	13,517/32,077/61,786
	TMH	100/45/24	0/0/0	0/0/0	0/0/0	13,321/24,561/27,295
	2SH	345/71/32	0/0/0	0/0/0	1/1/0	15,574/21,638/22,095
FirstName, *LastName*, *StreetAddress*, and *City*	BF	50,849/696/166	0/0/0	0/0/0	3/1/0	10,678/52,300/183
	TMH	605/279/43	0/0/0	0/0/0	1/1/0	45,878/48,634/31,828
	2SH	396/116/41	0/0/0	0/0/0	1/0/0	14,209/21,082/23,027

bourhood of a node will account for a considerably large change in the calculated features, which makes the matching of similar nodes more difficult. This can be observed throughout the experiments with all four attribute combinations.

Table 8.6 shows the PS values for the reidentification of entities. Note that mean and median risks are calculated based on only the encoded records that had at least one plaintext assignment, either correct or wrong. The encoded records that had no plaintext record assignments (and therefore PS value is 0) are not considered for mean and median risk calculations. Similar to attribute and identity reidentification results, the PS values for TMH and 2SH encodings, especially marketer risk values, are lower with three and four attribute combinations compared to one and two attribute combinations. Interestingly however, because of the way the attack matches nodes in a 1-to-1 manner using the SHM, SMM, or MWM methods as we discussed in Section 8.3.4.2, the attack was able to reidentify large number of real-world entities uniquely in certain instances as shown by the marketer risk. This is because most of the times a node in a graph corresponds to a single real-world entity especially when more than one QID value is encoded.

Finally, in Table 8.7 we show the runtime and memory consumption results for the different steps of our attack. As can be seen, feature generation is the most time consuming step while the reidentification of plaintext values took a maximum of around 1 hour complete. For databases with 500k and 1M records we could not run the attack using the same set of features and parameter settings used for the databases with 20k, 100k, and 200k records. With 500k records we stopped the attack after one week of running consuming more than 250GB (Gigabytes), whereas with 1M records we stopped the attack after five days consuming more than 320GB. However, the attack was able to finish running for 500k records when we used a smaller number of features and $s_m = 0.7$ and $c_m = 50$ parameter settings, resulting

Table 8.6: Maximum, minimum, mean, median, and marketer risk values of the records in the NCVR-100k encoded sensitive databases calculated based on probability of suspicion (PS) as we discussed in Section 3.7.1 on page 55.

Attribute combination	Encoding technique	Maximum risk	Minimum risk	100% / 75% / 50% Mean risk	Median risk	Marketer risk
FirstName	BF	1.0 / 1.0 / 1.0	0 / 0 / 0	0.13 / 0.12 / 0.11	0.01 / 0.01 / 0.01	0.092 / 0.002 / 0.003
	TMH	1.0 / 1.0 / 1.0	0 / 0 / 0	0.11 / 0.1 / 0.13	0.01 / 0.01 / 0.01	0.020 / 0.004 / 0.001
	2SH	1.0 / 1.0 / 1.0	0 / 0 / 0	0.11 / 0.12 / 0.09	0.01 / 0.01 / 0.01	0.035 / 0.007 / 0.002
FirstName and *LastName*	BF	1.0 / 1.0 / 1.0	0 / 0 / 0	0.97 / 0.96 / 0.93	1.0 / 1.0 / 1.0	0.394 / 0.019 / 0.012
	TMH	1.0 / 1.0 / 1.0	0 / 0 / 0	0.93 / 0.92 / 0.87	1.0 / 1.0 / 1.0	0.039 / 0.016 / 0.012
	2SH	1.0 / 1.0 / 1.0	0 / 0 / 0	0.92 / 0.89 / 0.87	1.0 / 1.0 / 1.0	0.193 / 0.068 / 0.019
FirstName, *LastName*, and *StreetAddress*	BF	1.0 / 1.0 / 1.0	0 / 0 / 0	1.0 / 0.8 / 1.0	1.0 / 0.8 / 1.0	0.469 / 0.003 / 0.003
	TMH	1.0 / 1.0 / 1.0	0 / 0 / 0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	0.001 / < 0.001 / < 0.001
	2SH	1.0 / 1.0 / 1.0	0 / 0 / 0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	0.003 / 0.001 / < 0.001
FirstName, *LastName*, *StreetAddress*, and *City*	BF	1.0 / 1.0 / 1.0	0 / 0 / 0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	0.508 / 0.007 / 0.002
	TMH	1.0 / 1.0 / 1.0	0 / 0 / 0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	0.006 / 0.003 / < 0.001
	2SH	1.0 / 1.0 / 1.0	0 / 0 / 0	1.0 / 1.0 / 1.0	1.0 / 1.0 / 1.0	0.004 / 0.001 / < 0.001

lower time and memory consumption compared to other experiments as can be seen in Table 8.7. Nevertheless, the reidentification accuracy results for that experiment was lower compared to others because most of the nodes in the generated graphs were not uniquely identifiable by their neighbourhood.

These experimental results of our proposed graph matching attack illustrate the limitations of encoding techniques such as BF, TMH, and 2SH, when used with certain parameter settings. Even though the TMH and 2SH encoding techniques have been proposed to overcome the privacy vulnerabilities of BF encoding, they are still vulnerable to privacy attacks, especially when only one or two attributes are encoded. With proper similarity adjustments, attribute values encoded using these methods can still be reidentified even when more than two attributes were encoded. Furthermore, our attack will work on any other encoding technique which calculates approximate similarities between record pairs.

8.6.4 Recommendations

Given the popularity of perturbation based encoding techniques to be used in real-world PPRL applications [10, 29, 153, 164], it is important to carefully consider and evaluate the privacy implications of such techniques before using them in practical applications. We recommend DOs to use our attack to evaluate the privacy guarantees of encoding techniques and the parameter settings they plan to use. Further, DOs can potentially use different hardening techniques proposed for BF encoding (as we discuss in Appendix A) to mitigate the risks of reidentification of QID values encoded in BFs. However, it is worth noting that no such hardening techniques are yet developed for both TMH and 2SH encoding.

Our attack has also emphasised the need of further research into developing more advanced encoding techniques to be used in PPRL, where such techniques still need to be scalable to large databases and provide high linkage quality. We recommend to

Table 8.7: Time (in seconds) and memory (in Megabytes) consumption analysis of different steps of the proposed attack (discussed in Sections 8.3.2 to 8.3.5) for different databases averaged over other parameter settings. For each step we refer to its corresponding section where we describe that step.

Number of records	Time (sec) / Mmory (MB) Graph generation Section 8.3.2	Feature calculation Section 8.3.3	Node matching Section 8.3.4	Plaintext reidentification Section 8.3.5
20k	126 / 6,300	1,757 / 7,035	272 / 7,401	13 / 7,401
100k	2,656 / 17,785	117,490 / 25,703	3,080 / 27,284	720 / 27,276
200k	5,251 / 20,953	142,862 / 27,373	13,992 / 29,637	3,664 / 29,646
500k	63,627 / 25,242	772 / 27,744	936 / 27,951	545 / 27,963
1M	Experiment terminated due to high memory consumption			

develop encoding techniques that do not provide encoding similarities very similar to the corresponding plaintext similarities and also do not yield a linear relationship between encoded and plaintext similarities. The proposed graph matching attack would be difficult to conduct on such encoding techniques for PPRL because of the distorted similarities.

8.7 Summary

We have proposed a novel graph matching attack on privacy-preserving record linkage (PPRL) that, unlike earlier attack methods for PPRL, can be applied on any PPRL technique that calculates similarities between encoded values. Our attack reidentifies sensitive values in an encoded database from a known plaintext database by using graph matching. We successfully applied our attack on three encoding techniques, popular Bloom filters [177], tabulation min-hashing [186] and two-step hashing [158]. Given the increasing demand for such perturbation based encoding techniques in real-world PPRL applications, it is essential to investigate the privacy limitations of such techniques.

Chapter 9

A Comparative Evaluation of Attacks on Privacy-Preserving Record Linkage

In Chapters 6 to 8 we proposed three novel attack methods on different privacy-preserving record linkage techniques to identify their vulnerabilities. In this chapter, we evaluate and compare these three attack methods with four baseline attack methods using three databases and diverse set of parameter settings in order to understand their performance and limitations in real-world linkage scenarios. In Section 9.1 we first provide a brief description about the three attack methods we proposed in this thesis, and in Section 9.2 we then introduce the baseline attack methods used for the comparative evaluation and why they are being selected. In Section 9.3 we describe the databases and parameter settings we used for the analysis of the attacks. In Section 9.4 we then discuss the evaluation criteria we use, and in Sections 9.5, and 9.6 we illustrate the obtained results and discuss major findings. In Section 9.7 we then list a set of recommendations that should be considered when conducting real-world PPRL projects. Finally, in Section 9.8 we provide a summary of the chapter.

9.1 Introduction

As we discussed in Chapters 1 and 4, existing attacks on privacy-preserving record linkage (PPRL) have several limitations, including impractical assumptions about the prior knowledge of the adversary, limited scope, and excessive computational cost. In this thesis we proposed three novel attack methods to overcome these limitations while achieving high reidentification accuracy.

The first attack method we proposed (in Chapter 6) is on multiple match-key (MMK) encoding technique proposed by Randall et al. [163]. The idea of this attack is to align frequent plaintext quasi-identifying (QID) attribute combinations (called plaintext match-keys) with frequent encoded QID combinations (called encoded match-keys) from the plaintext and encoded databases, respectively. This

attack employs a set of statistical correlation measures to compare the frequency distributions of encoded match-key values with the frequency distributions of plaintext match-key values. We evaluated this attack assuming different levels of the assumed knowledge of the adversary about the QIDs used for encoded match-keys. The adversary does not require any other information about the encoding parameters to conduct this attack. This attack method is specific to MMK encoding due to the method of utilising one-way hash encoding functions. However, this attack could potentially be applied on any other encoding technique of similar nature. We refer to this attack method as **FrqVid** in this chapter.

The second attack method we proposed (in Chapter 7) is an extension of the cryptanalysis attack on Bloom filter (BF) encoding [177] proposed by Christen et al. [43]. In this attack we first use a maximal frequent pattern-mining based approach to identify and align frequent q-grams from a plaintext database with frequent sets of bit positions from an encoded database of BFs. We extended the initial version of the attack using a language model based approach, and a plaintext/BF partitioning based approach to further identify non-frequent q-grams and their corresponding sets of bit positions. Unlike previous attacks on BF encoding, our attack does not require specific knowledge about encoding parameters and it will work even when all BFs in the encoded database are unique. We refer to this attack method as **PtMinVid** in this chapter.

The third attack method we proposed (in Chapter 8) exploits the similarity neighbourhood of records from the encoded and plaintext databases to reidentify encoded QID attribute values. Two similarity graphs are first built using the plaintext and the encoded databases, where nodes in each graph represent the records in the corresponding database while edges represent the similarities calculated between these records. The idea of the attack is to align nodes in the encoded graph with nodes in the plaintext graph by comparing their similarity neighbourhoods using a set of node specific features. This alignment allows the attack to reidentify encoded QID values with high accuracy, as we showed in Chapter 8. Unlike the previous two attack methods we proposed, this attack can be applied on any PPRL encoding technique that calculates approximate similarities between encoded records. To illustrate this capability, we evaluated our graph attack using three encoding techniques BF [177], tabulation min-hashing (TMH) [186], and two-step hashing (2SH) [158]. We refer to this attack method as **GrpVid** in this chapter.

9.2 Baseline Attack Methods

We used four state-of-the-art attack methods as baselines to be compared with our proposed attack methods. The methodologies of all four of these attack methods are discussed in detail in Chapter 4. These four attack methods are,

1. Optimisation algorithm based cryptanalysis on BF encoding by Kroll and Steinmetzer [119] (**OptKro**) as discussed in Section 4.2.3 on page 71.

2. Frequency based cryptanalysis on BF encoding by Christen et al. [42] (**FrqChr**) as discussed in Section 4.2.4 on page 72.

3. Extended frequency based cryptanalysis on BF encoding by Christen et al. [41] (**ExFrqChr**) as discussed in Section 4.2.5 on page 74.

4. Graph traversal based cryptanalysis on BF encoding by Mitchell et al. [139] (**GrpMit**) as discussed in Section 4.2.6 on page 76.

We selected these attack methods because they cover a wide range of dimensions under the three aspects adversarial, technical, and practical as we detailed in our taxonomy of attacks on PPRL in Chapter 3.

- **Adversarial aspects**: The **OptKro**, **FrqChr**, and **ExFrqChr** attacks assume both inside and outside adversaries since the attack can be performed by either of those adversaries whereas the **GrpMit** attack assumes an inside adversary. The **OptKro** attack assumes that the adversary knows the number of hash functions used for BF encoding, the q-gram length, and the QID value(s) used for the encoding. This attack also assumes that the encoding of QID values was conducted using the double hashing (DH) method [52]. The **GrpMit** attack requires the adversary to have complete knowledge of all parameters used in the BF encoding process, including the shared secret keys, if any. On the other hand, the **FrqChr** and **ExFrqChr** attacks only assume that the adversary has access to a plaintext database and knows about the q-gram length and the QID value(s) used for the encoding.

- **Technical aspects**: The **OptKro**, **FrqChr**, and **ExFrqChr** are cryptanalysis attacks where they exploit the frequency distribution of encoded QID values. Apart from the frequency these attacks also exploits co-occurence and length vulnerabilities as we discussed in Chapter 3. The **GrpMit** attack, on the other hand, is a dictionary based attack and exploits the co-occurrence vulnerability of encoded QID values. All four of the attacks are proposed for BF encoding.

- **Practical aspects**: The **OptKro** and **GrpMit** attacks are evaluated using only one database whereas the **FrqChr** and the **ExFrqChr** attacks are evaluated using two and three databases, respectively. In terms of QID value(s) used for the evaluation, the **OptKro**, **FrqChr**, and **GrpMit** attacks have used first name and last name attributes, whereas the **ExFrqChr** attack has used first name, last name, and city attributes. All four attacks are fully automated.

9.3 Databases and Parameter Settings

For the comparative evaluation we used two real-world databases and two synthetic databases. First, we used two synthetic European (EURO) databases[1], where one

[1] https://ec.europa.eu/eurostat/cros/content/job-training_en

Table 9.1: Percentages of matching records and changes in QID attribute values in all databases pairs.

Database pair	Number of records	Percentage of matches	Percentage of changed records			
			FirstName	*LastName*	*StreetAddress*	*City / Zipcode*
EURO-CS / EURO-CS	25,343 each	100%	0%	0%	0%	0%
EURO-CS / EURO-CIS	25,343 each	94.87%	33.87%	29.36%	10.54%	7.16%
NCVR-14e / NCVR-14p	222,257 / 222,202	66.6%	7.91%	4.99%	86.02%	76.98%
NCVR-21e / NCVR-20p	7,189,744 / 7,544,296	99.14%	0.03%	0.11%	1.06%	0.47%
NCVR-21e / NCVR-19p	7,189,744 / 6,697,656	84.28%	0.47%	1.38%	17.34%	8.76%
NCVR-21e / NCVR-13p	7,189,744 / 7,251,658	63.61%	1.16%	5.99%	42.14%	22.86%
MVR-16e / MVR-14p	200,000 each	90.0%	1.73%	21.14%	26.0%	3.35%

is a census (CS) database generated to represent real observations of the decennial census and the second is a customer information system (CIS) database generated to represent fictional observations from administrative data. This database pair included 94.9% of true matches, and from those 57.5% of record pairs had at least one change in any of the QID values *FirstName*, *LastName*, *StreetAddress*, and *Zipcode*. Each database contained 25,343 records. For one set of experiments we used EURO-CS database as both the encoded ($\mathbf{D}^e$) and plaintext ($\mathbf{D}^p$) databases and for the other set we used EURO-CS as the encoded database $\mathbf{D}^e$ and EURO-CIS as the plaintext database $\mathbf{D}^p$.

Second, we used a North Carolina Voter Registration (NCVR) database[2] where we selected two snapshots, one from June 2014 as the encoded database $\mathbf{D}^e$ and another from April 2014 as the plaintext database $\mathbf{D}^p$ to create a database pair which consists of records of the same voters but had names and/or addresses changed over time. This database pair included 66.6% of true matches, and from those true matches 98.9% of record pairs had at least one change in any of the selected QID values *FirstName*, *LastName*, *StreetAddress*, and *City*. The resulting two databases had 222,257 ($\mathbf{D}^e$) and 222,202 ($\mathbf{D}^p$) records and we refer to those database pair as NCVR-14e and NCVR-14p, respectively. We then used four NCVR snapshots collected in February 2021, December 2020, February 2019, and February 2013, where we used the snapshot from February 2021 (NCVR-21e) as the encoded database $\mathbf{D}^e$ and the others (NCVR-20p, NCVR-19p, and NCVR-13p) as the plaintext databases $\mathbf{D}^p$. The selected databases contained between 6 to 7.5 million records in each and they had different percentages of missing and/or changed values over time in selected QIDs.

[2]https://dl.ncsbe.gov/

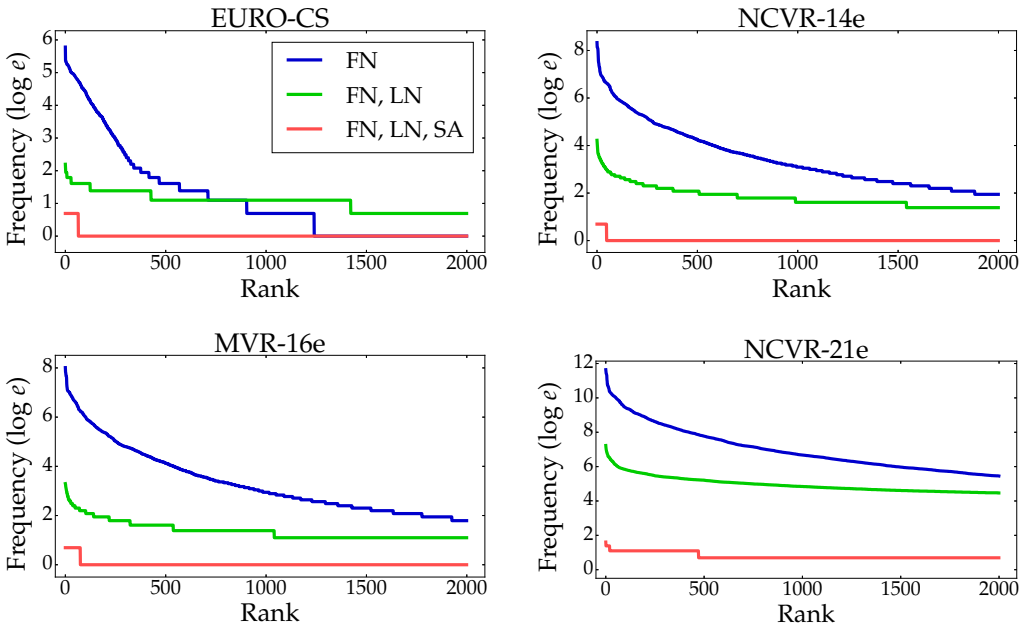


Figure 9.1: Frequency distributions of the top 2,000 QID value combinations in encoded databases. QIDs used are *FirstName* (FN), *LastName* (LN), and *StreetAddress* (SA). With four attribute combination (with *City* or *Zipcode*) the frequency distribution is the same as with three attribute combination.

The third database is the Michigan voter registration (MVR) database[3] where we selected two snapshots, one from September 2016 as the encoded database $\mathbf{D}^e$ and another from September 2014 as the plaintext database $\mathbf{D}^p$ to create a database pair. This database pair included 90% true matches between them, and from those 50% of record pairs had at least one change in any of the QID values *FirstName*, *LastName*, *StreetAddress*, and *City*. The resulting two databases had 200,000 records in each and we refer to those database pair as MVR-16e ($\mathbf{D}^e$) and MVR-14p ($\mathbf{D}^p$). In Table 9.1 we show the percentages of changes in each QID attribute in all databases pairs discussed above. As can be expected, the number of changes monotonically increases as the time differences between database pairs increase.

In Figures 9.1 and 9.2, we analyse the frequency distributions of QID values as well as bigrams, with $l_q = 2$, in all instances of encoded databases. As can be seen, with individual QIDs the frequency distribution of values is more distinct and therefore some of these values (with distinct frequencies) can potentially be uniquely identified. However, with three and four combination of QIDs the frequency distribution of values becomes more uniform (the frequencies of most of the values are 1) making them difficult to be uniquely identified. This is because when multiple QIDs are combined into a single value (such as *FirstName+LastName+StreetAddress* where '+' denotes the concatenation of strings), that value will highly likely to be unique

[3] http://michiganvoters.info/

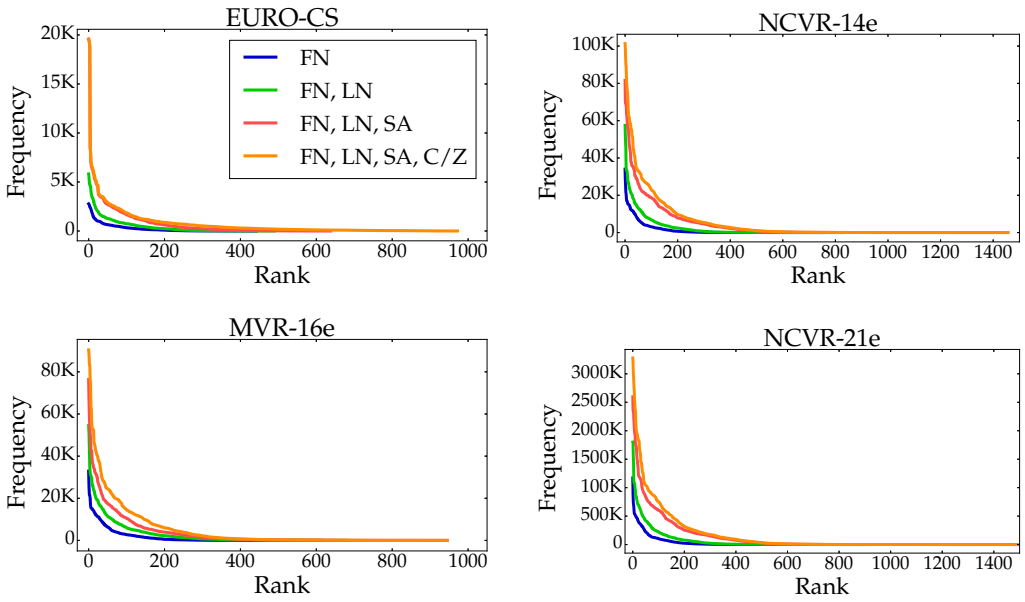


Figure 9.2: Frequency distributions of bigrams extracted from different QID value combinations in encoded databases. QIDs used are *FirstName* (FN), *LastName* (LN), *StreetAddress* (SA), and *City*/*Zipcode* (C/Z).

among all possible values [190]. However, even with three and four combinations of QIDs the frequencies of q-grams are distinct and therefore they can be uniquely identified as can be seen in the Figure 9.2. We used different attribute combinations of the QIDs *FirstName* (FN), *LastName* (LN), and*StreetAddress* (SA), *City* (C), and *Zipcode* (Z) to generate different instances of encoded databases using a set of encoding parameters as discussed below.

We used the following parameter settings for the encoding of databases. Following earlier work [27, 64, 177], for BF encoding we used the settings $l_q = 2$ q-gram length, $l = 1,000$ BF length, $k = [15, opt]$ number of hash functions, and with and without padding. BFs were encoded using the Cryptographic Long-term Key (CLK) approach [178]. We also experimented with two BF hardening techniques, window based XORing and re-sampling based XORing [161] where we used a window size $w = 2$ for window based XORing. Following Smith [186], for TMH encoding we used $l_q = 2$ q-gram length and $c = [4, 8]$ look-up tables per min-hash signature, where in each look-up table we used bit keys of length $k = [4, 8]$ to refer to 32 and 64 bit random bit strings, respectively. Using the obtained min-hash signatures, bit arrays of length $l = 1,000$ were generated for each encoded value [186]. Following Ranbaduge et al. [158], for 2SH encoding we used the parameter settings, $l_q = 2$ q-gram length, $l = 1,000$ number of hash columns, $k = [10, 15]$ number of hash functions, and with and without padding. Following Randall et al. [163], for MMK encoding we used the five attributes *FirstName*, *LastName*, *StreetAddress*, *City*, and *Zipcode* to generate match-keys that provided the highest linkage quality. The lengths of the match-keys (the number of QID attributes used) ranged from two to four.

Table 9.2: Attribute reidentification accuracy results for all seven attack methods with the EURO database pairs (described in Section 9.3).

Attack method	QID attribute combination	EURO (CS/CS) / (CS/CIS) 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
OptKro	FN and LN	747 / 5058	0 / 26	1144 / 3216	0 / 188	5 / 305
	FN, LN, and SA	Experiments terminated due to high time consumption				
FrqChr	FN and LN	0 / 0	0 / 0	1 / 1	1 / 1	0 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	11,877 / 5,260	0 / 0	3,586 / 3,257	0 / 0	0 / 1
	FN, LN, and SA	11,273 / 4,925	0 / 0	35 / 25	0 / 0	0 / 0
ExFrqChr	FN and LN	1 / 0	0 / 0	1 / 0	1/ 0	0 / 2
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	4 / 1	0 / 0	0 / 0	0 / 0	29 / 21
	Attr	4 / 1	0 / 0	0 / 0	0 / 0	29 / 21
	Dom	4 / 1	0 / 0	0 / 0	0 / 0	29 / 22
PtMinVid	FN and LN	914 / 289	0 / 0	789 / 686	0 / 0	255 / 926
	FN, LN, and SA	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
GrpVid (BF)	FN and LN	18,750 / 160	23 / 1	0 / 0	0 / 0	0 / 6,879
	FN, LN, and SA	25,223 / 14	0 / 0	0 / 0	0 / 0	53 / 1,937
GrpVid (TMH)	FN and LN	12,642 / 688	21 / 6	0 / 0	0 / 0	4,673 / 6,979
	FN, LN, and SA	4,612 / 158	0 / 0	0 / 0	0 / 0	16,601 / 752
GrpVid (2SH)	FN and LN	16,905 / 758	23 / 4	0 / 0	0 / 0	833 / 7,025
	FN, LN, and SA	5,825 / 520	0 / 0	0 / 0	0 / 0	16,906 / 12,885

For the **OptKro** attack method we encoded the sensitive databases using the DH method [52] since the attack will only work when the q-grams are hashed using the DH. For the **FreqChr**, **ExFrqChr**, and **GrpMit**, attack methods we used the parameters recommended in their respective publications [41, 42, 139]. For all the parameter settings specific to our proposed three attack methods, we used the settings that provided good reidentification results in our previously conducted experiments as we discussed in Chapters 6 to 8.

9.4 Evaluation Criteria

Following the taxonomy of attacks on PPRL as we discussed in Chapter 3, we measured both the attribute and the identity reidentification accuracy for each of the attacks. The attribute reidentification accuracy measures how accurate an attack is on reidentifying encoded QID values, whereas identity reidentification measures how accurately an attack reidentifies actual real-world entities using the identified QID values. Similar to the previous chapters we calculated the number of 1-to-1, 1-to-*many*, *many*-to-1, and *many*-to-*many* attribute and identity reidentifications. We also calculated the probability of suspicion (PS) values for each record in the encoded database $\mathbf{D}^e$, as we discussed in Section 3.7.1. The purpose behind this experimental evaluation is to compare the performance of the proposed attack methods with the baseline attack methods, and analyse how certain real-world aspects, such as changes and errors in QID values, affect the accuracy of the attacks.

Table 9.3: Identity reidentification accuracy results for all seven attack methods with the EURO database pairs (described in Section 9.3).

Attack method	QID attribute combination	EURO (CS/CS) / (CS/CIS) 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
OptKro	FN and LN	645 / 4884	0 / 0	0 / 0	1020 / 1	0 / 808
	FN, LN, and SA	Experiments terminated due to high time consumption				
FrqChr	FN and LN	0 / 0	0 / 0	0 / 0	1 / 1	0 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	11,877 / 3,942	0 / 735	5 / 924	3,581 / 2135	0 / 782
	FN, LN, and SA	11,273 / 4,910	0 / 8	0 / 16	35 / 9	0 / 7
ExFrqChr	FN and LN	0 / 0	0 / 0	0 / 0	2 / 0	0 / 2
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	0 / 0	0 / 0	0 / 0	4 / 1	29 / 21
	Attr	0 / 0	0 / 0	0 / 0	4 / 1	29 / 21
	Dom	0 / 0	0 / 0	0 / 0	4 / 1	29 / 22
PtMinVid	FN and LN	833 / 214	17 / 74	47 / 64	700 / 705	5 / 134
	FN, LN, and SA	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
GrpVid (BF)	FN and LN	14,470 / 87	0 / 12	0 / 14	4,303 / 41	0 / 6,885
	FN, LN, and SA	25,158 / 19	0 / 0	0 / 0	65 / 0	53 / 1,934
GrpVid (TMH)	FN and LN	9,292 / 367	0 / 32	0 / 50	3371 / 219	4,673 / 6,998
	FN, LN, and SA	4,565 / 189	0 / 0	0 / 0	48 / 0	16,601 / 746
GrpVid (2SH)	FN and LN	12,840 / 382	0 / 37	0 / 56	4,092 / 245	833 / 7,060
	FN, LN, and SA	5,766 / 610	0 / 0	0 / 0	59 / 6	16,906 / 12,789

9.5 Experimental Results

In Tables 9.2 and 9.3 we show the attribute and identity reidentification results of different attack methods, when using the EURO database pairs as discussed in Section 9.3. As can be seen, apart from the **GrpMit** attack, our attacks performed better compared to the other baseline attacks. Both the **FrqChr** and **ExFrqChr** attack methods did not provide results when using three QID attribute combinations. This is because with three QID attributes none of the encoded values had the minimum frequency limit and/or the minimum frequency difference between two values required by these attacks. The **OptKro** attack also worked only on two QID attribute combinations. With three QID attribute combination we stopped the experiments after running for more than 750 hours. With the EURO CS/CIS database pair both the **OptKro** and **GrpMit** attacks obtained better results compared to the **PtMinVid** and **GrpVid** attacks. However, with EURO CS/CS database pair both the **PtMinVid** and **GrpVid** attacks obtained a large amount of reidentifications compared to the **OptKro** attack.

With three attribute combinations the reidentification results for our attacks are also comparatively low. With the **PtMinVid** attack method this is because the attack could not find any frequent q-grams that have the required minimum frequency difference between them. On the other hand, with the **GrpVid** attack method the results for three QID attribute combination are low because of the dissimilar neighbourhoods of nodes when more QID values are encoded together, as we discussed in

Table 9.4: Attribute reidentification accuracy results for six attack methods with the NCVR-14 and MVR database pairs (described in Section 9.3).

Attack method	QID attribute combination	NCVR (14e/14p) / MVR (16e/14p) 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FrqChr	FN and LN	13 / 1	2 / 1	0 / 0	1 / 1	1 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	80,472 / 313	0 /0	0 / 17	11,493 / 0	30 / 150
	FN, LN, and SA	Experiments terminated due to high memory consumption				
ExFrqChr	FN and LN	10 / 1	0 / 1	6 / 0	2 / 1	0 / 0
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	2 / 1	3 / 0	7 / 0	0 / 2	8 / 40
	Attr	0 / 2	3 / 0	10 / 14	2 / 2	20 / 76
	Dom	0 / 4	2 / 0	9 / 13	2 / 1	15 / 71
PtMinVid	FN and LN	16,852 / 33,780	0 / 0	4,070 / 5,965	0 / 0	3,184 / 0
	FN, LN, and SA	2,281 / 42484	0 / 0	96 / 5428	0 / 0	17,562 / 10137
GrpVid (BF)	FN and LN	684 / 1,268	48 / 5	58 / 0	141 / 0	887 / 147
	FN, LN, and SA	9 / 22	0 / 0	0 / 0	0 / 0	2 / 12
GrpVid (TMH)	FN and LN	5,304 / 1,584	356 / 7	340 / 0	1,007 / 0	3,614 / 537
	FN, LN, and SA	10 / 37	0 / 0	0 / 0	0 / 0	3,612 / 38
GrpVid (2SH)	FN and LN	2,804 / 2,404	178 / 8	192 / 0	516 / 0	5,402 / 114
	FN, LN, and SA	4 / 828	0 / 2	0 / 0	0 / 0	385 / 8

Table 9.5: Identity reidentification accuracy results for six attack methods with the NCVR-14 and MVR database pairs (described in Section 9.3).

Attack method	QID attribute combination	NCVR (14e/14p) / MVR (16e/14p) 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FrqChr	FN and LN	0 / 0	10 / 2	0 / 0	1 / 1	1 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	74,883 / 312	3,426 / 6	3,406 / 6	7,797 / 11	2,472 / 145
	FN, LN, and SA	Experiments terminated due to high memory consumption				
ExFrqChr	FN and LN	0 / 0	8 / 1	0 / 0	9 / 2	0 / 0
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	0 / 0	0 / 0	0 / 0	11 / 5	47 / 49
	Attr	0 / 0	0 / 0	0 / 0	287 / 651	628 / 1,370
	Dom	0 / 0	0 / 0	0 / 0	287 / 641	628 / 1,380
PtMinVid	FN and LN	15,915 / 32,353	1,156 / 1,166	1,482 / 1,788	3,033 / 4,522	1,667 / 1,286
	FN, LN, and SA	1,043 / 42,583	215 / 1,774	239 / 2,744	1,000 / 4,129	12,992 / 4,184
GrpVid (BF)	FN and LN	684 / 1,000	48 / 35	58 / 40	141 / 183	887 / 147
	FN, LN, and SA	8 / 22	0 / 0	0 / 0	2 / 0	2 / 12
GrpVid (TMH)	FN and LN	5,304 / 1,274	356 / 39	340 / 37	1,007 / 224	3,614 / 539
	FN, LN, and SA	10 / 38	0 / 0	0 / 0	0 / 0	3,612 / 38
GrpVid (2SH)	FN and LN	2,804 / 1,933	178 / 57	192 / 61	516 / 334	5,402 / 115
	FN, LN, and SA	4 / 819	0 / 0	0 / 0	0 / 11	385 / 8

Chapter 8. The **FreqVid** attack method also provided low results, but it is still better than the results obtained by the **FrqChr** and **ExFrqChr** attack methods.

Tables 9.4 and 9.5 show the attribute and identity reidentification results obtained for the NCVR-14 and MVR database pairs. These results are comparable to the results of the EURO database pairs. Both the **FrqChr** and **ExFrqChr** attack methods again did not provide results for three QID attribute combinations, and when com-

Table 9.6: Attribute reidentification accuracy results for three attack methods with the large NCVR database pairs (described in Section 9.3).

Attack method	QID attribute combination	NCVR (21e/20p) / (21e/19p) / (21e/13p) 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FrqChr	FN and LN	5 / 5 / 2	2 / 1 / 0	0 / 0 / 0	2 / 2 / 1	0 / 0 / 1
	FN, LN, and SA	Not enough frequency information found				
ExFrqChr	FN and LN	4 / 2 / 0	1 / 0 / 0	3 / 0 / 0	2 / 1 / 0	0 / 0 / 0
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	8 / 0 / 1	1 / 0 / 0	19 / 10 / 5	5 / 5 / 9	34 / 43 / 41
	Attr	0 / 0 / 1	0 / 0 / 0	17 / 6 / 3	5 / 6 / 9	34 / 36 / 33
	Dom	0 / 0 / 1	0 / 0 / 0	15 / 5 / 3	5 / 5 / 7	30 / 32 / 29

Table 9.7: Identity reidentification accuracy results for three attack methods with the large NCVR database pairs (described in Section 9.3).

Attack method	QID attribute combination	NCVR (21e/20p) / (21e/19p) / (21e/13p) 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
FrqChr	FN and LN	0 / 0 / 0	5 / 4 / 2	0 / 0 / 0	2 / 2 / 1	0 / 0 / 1
	FN, LN, and SA	Not enough frequency information found				
ExFrqChr	FN and LN	0 / 0 / 0	4 / 2 / 0	0 / 0 / 0	2 / 2 / 0	0 / 0 / 0
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	0 / 0 / 0	0 / 0 / 0	3 / 0 / 0	17,539 / 5,969 / 8,551	22,545 / 31,372 / 24,661
	Attr	0 / 0 / 0	0 / 0 / 0	3 / 0 / 0	17,354 / 5,940 / 8,455	22,893 / 31,692 / 24,679
	Dom	0 / 0 / 0	0 / 0 / 0	2 / 0 / 0	14,164 / 3,225 / 5,671	23,745 / 30,471 / 26,347

pared to these two attacks the **FreqVid** attack method provided better results. The **GrpMit** attack was also not able to run on three QID attribute combinations due to high memory consumption. We stopped the experiments on NCVR-14 and MVR database pairs after consuming more than 400GB and 250GB memory, respectively. We excluded the **OptKro** attack from these tables because the experiments did not finish running for both attribute combinations. This is because with more plaintext q-grams and encoded atoms in two matrices, the optimisation process of these matrices takes a long time. For instance, with the NCVR-14 database pair and two QID attribute combination, the **OptKro** attack experiments consumed more than 820 hours before we terminated it.

Unlike with the EURO databases (especially with the EURO CS/CS database pair), with the NCVR-14 and MVR database pairs the **PtMinVid** attack gave better results when compared to the **GrpVid** attack. This is because, as we mentioned in Section 9.3, the NCVR-14 and MVR database pairs only had 66.6% and 90% matching records, respectively, while the EURO CS/CS and EURO CS/CIS database pairs had 100% and 94.9% matching records. Since more non-matching records will lead to dissimilar neighbourhoods for nodes in encoded and plaintext graphs, the **GrpVid** attack will not yield good results with such databases. However, since the **PtMinVid** attack method exploits frequent q-grams and the corresponding frequent bit patterns, despite having non-matching records the attack can still yield good results.

Table 9.8: Maximum, minimum, mean, median, and marketer risk values of the records in the EURO database pairs calculated based on probability of suspicion (PS) as we discussed in Section 3.7.1 on page 55.

Attack method	QID attribute combination	EURO (CS/CS) / (CS/CIS) Maximum risk	Minimum risk	Mean risk	Median risk	Marketer risk
OptKro	FN and LN	1.0 / 1.0	0 / 0	0.08 / 0.52	0.01 / 0.5	0.030 / 0.193
	FN, LN, and SA	Experiments terminated due to high time consumption				
FrqChr	FN and LN	0.08 / 0.14	0 / 0	0.05 / 0.14	0.06 / 0.14	0 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	1.0 / 1.0	0 / 0	0.75 / 0.69	1.0 / 0.50	0.468 / 0.191
	FN, LN, and SA	1.0 / 1.0	0 / 0	1.0 / 1.0	1.0 / 1.0	0.444 / 0.194
ExFrqChr	FN and LN	0.12 / < 0.01	0 / 0	0.12 / < 0.01	0.11 / < 0.01	0 / 0
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	0.13 / 0.09	0 / 0	0.09 / 0.09	0.09 / 0.09	0 / 0
	Attr	0.13 / 0.09	0 / 0	0.09 / 0.09	0.09 / 0.09	0 / 0
	Dom	0.13 / 0.09	0 / 0	0.09 / 0.09	0.09 / 0.09	0 / 0
PtMinVid	FN and LN	1.0 / 1.0	0 / 0	0.14 / 0.1	0.07 / 0.04	0.032 / 0.013
	FN, LN, and SA	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
GrpVid (BF)	FN and LN	1.0 / 1.0	0 / 0	0.75 / 0.74	1.0 / 1.0	0.582 / 0.004
	FN, LN, and SA	1.0 / 1.0	0 / 0	0.99 / 1.0	1.0 / 1.0	0.991 / < 0.001
GrpVid (TMH)	FN and LN	1.0 / 1.0	0 / 0	0.71 / 0.73	1.0 / 0.75	0.366 / 0.016
	FN, LN, and SA	1.0 / 1.0	0 / 0	0.99 / 1.0	1.0 / 1.0	0.180 / 0.007
GrpVid (2SH)	FN and LN	1.0 / 1.0	0 / 0	0.74 / 0.66	1.0 / 0.5	0.506 / 0.017
	FN, LN, and SA	1.0 / 1.0	0 / 0	0.99 / 0.99	1.0 / 1.0	0.227 / 0.024

In Tables 9.6 and 9.7 we illustrate the attribute and identity reidentification results obtained for the large NCVR database pairs (NCVR 21e/20p, 21e/19p, and 21e/13p). As can be seen, in terms of 1-to-1 and 1-to-*many* matches the **FrqChr** and **ExFrqChr** attacks provided better results compared to the **FreqVid** attack. However, overall the **FreqVid** attack gave a higher number of total reidentifications compared to the **FrqChr** and **ExFrqChr** attack methods. With identity reidentification the **FreqVid** attack had a large number of *many*-to-*many* reidentifications because almost all of the identified QID attribute values had high frequencies where they occur in many records. On the other hand, since the **FrqChr** and **ExFrqChr** attacks are based on simple frequency alignment of QID attribute values, when two or more QID attributes are used for encoding, such a simple alignment will not be accurate enough to obtain good reidentification results.

With the large NCVR database pairs we were only able to obtain results for the **FrqChr**, **ExFrqChr**, and **FreqVid** attack methods. All the other attack methods had high time and/or memory consumption which are not feasible in real-world scenarios. The **OptKro** attack consumed a large amount of time to optimise the encoded and plaintext matrices. The **GrpMit** attack had both high time and memory consumption. The graph traversal process consumes time and with more attributes encoded into BFs there will potentially be a large number of simple or edge-disjoint walks to consider, which consumes both memory and time. For instance, with three attribute combination (FN, LN, and SA) certain q-grams sets assigned to encoded records had more than 50,000 simple walks. The **PtMinVid** attack had an excessive

Table 9.9: Maximum, minimum, mean, median, and marketer risk values of the records in the NCVR-14 and MVR database pairs calculated based on probability of suspicion (PS) as we discussed in Section 3.7.1 on page 55.

Attack method	QID attribute combination	NCVR (14e/14p) / MVR (16e/14p) Maximum risk	Minimum risk	Mean risk	Median risk	Marketer risk
FrqChr	FN and LN	0.02 / 0.04	0 / 0	0.02 / 0.02	0.02 / 0.02	0 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	1.0 / 1.0	0 / 0	0.85 / 0.94	1.0 / 1.0	0.352 / 0.002
	FN, LN, and SA	Experiments terminated due to high memory consumption				
ExFrqChr	FN and LN	0.03 / 0.04	0 / 0	0.02 / 0.03	0.02 / 0.04	0 / 0
	FN, LN, and SA	Not enough frequency information found				
FrqVid	**Comb**	0.14 / 0.2	0 / 0	0.13 / 0.2	0.13 / 0.2	0 / 0
	Attr	0.02 / 0.06	0 / 0	0.01 / 0.01	0.02 /0.01	0 / 0
	Dom	0.02 / 0.14	0 / 0	0.01 / 0.01	0.02 / 0.01	0 / 0
PtMinVid	FN and LN	1.0 / 1.0	0 / 0	0.73 / 0.79	1.0 / 1.0	0.093 / 0.187
	FN, LN, and SA	1.0 / 1.0	0 / 0	0.39 / 0.85	0.25 / 1.0	0.013 / 0.244
GrpVid (BF)	FN and LN	1.0 / 1.0	0 / 0	0.75 / 0.77	1.0 / 1.0	0.003 / 0.005
	FN, LN, and SA	1.0 / 1.0	0 / 0	1.0 / 1.0	1.0 / 1.0	< 0.001 / < 0.001
GrpVid (TMH)	FN and LN	1.0 / 1.0	0 / 0	0.76 / 0.78	1.0 / 1.0	0.025 / 0.007
	FN, LN, and SA	1.0 / 1.0	0 / 0	1.0 / 1.0	1.0 / 1.0	< 0.001 / < 0.001
GrpVid (2SH)	FN and LN	1.0 / 1.0	0 / 0	0.74 / 0.82	1.0 / 1.0	0.013 / 0.01
	FN, LN, and SA	1.0 / 1.0	0 / 0	1.0 / 1.0	1.0 / 1.0	0.001 / 0.004

runtime due to having more plaintext/BF partitions to consider in the third step of the attack. The **GrpVid** attack had high memory consumption because with very large databases the generated plaintext and encoded similarity graphs will become more dense, leading to very large neighbourhoods.

In Tables 9.8 and 9.9 we show the PS values [200] for the reidentification of entities when using EURO, NCVR, and MVR database pairs. Note that as with previous chapters we calcualted the mean and median risks based on only the encoded records that had at least one plaintext assignment, either correct or wrong. As can be seen, none of the attack methods were able to assign plaintext entities to all the entities in the encoded database. Therefore, all the attacks had 0 minimum risk indicating that there are encoded records with no assignments.

When considering all the database pairs, the **GrpMit** and the **GrpVid** attack methods gave the best PS results having a maximum PS value of 1.0 and a considerably large number of such records (with PS value 1.0) leading to high marketer risk. The **OptKro** and **PtMinVid** attacks provided the second best PS results because with the EURO database pairs the **PtMinVid** attack had comparatively low attribute and identity reidentification results. However, with the NCVR-14 and MVR database pairs the **OptKro** attack did not have any results at all due to the reasons we discussed above. With the NCVR-14 and MVR database pairs the **PtMinVid** attack gave better results than the **GrpVid** attack. The **FrqVid** attack had much lower results compared to above three attacks and the **FrqChr** and **ExFrqChr** attacks had the worst PS results. All three of these attack methods (**FrqVid**, **FrqChr**, and **ExFrqChr**) had no encoded records (entities) with 1-to-1 matches and therefore the maximum PS value is lower than 1.0.

Table 9.10: Attribute reidentification accuracy results for attack methods on BF encoding with the EURO (CS/CS) database pair when two hardening methods, window based XORing [161] and re-sampling based XORing [161] are applied.

Attack method	QID attribute combination	Window based XORing / Re-sampling based XORing 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
OptKro	FN and LN	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
	FN, LN, and SA	Experiments terminated due to high time consumption				
FrqChr	FN and LN	0 / 0	0 / 0	0 / 0	1 / 1	0 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	5,193 / 0	0 / 0	1,570 / 0	0 / 0	0 / 0
	FN, LN, and SA	4,690 / 0	0 / 0	16 / 0	0 / 0	0 / 0
ExFrqChr	FN and LN	2 / 2	1 / 1	1 / 0	1 / 0	0 / 0
	FN, LN, and SA	Not enough frequency information found				
PtMinVid	FN and LN	894 / 0	0 / 0	781 / 0	0 / 0	259 / 0
	FN, LN, and SA	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
GrpVid (BF)	FN and LN	16,914 / 16,151	51 / 44	0 / 0	0 / 0	0 / 3
	FN, LN, and SA	25,234 / 24,569	1 / 1	0 / 0	0 / 0	0 / 1

Table 9.11: Identity reidentification accuracy results for attack methods on BF encoding with the EURO (CS/CS) database pair when two hardening methods, window based XORing [161] and re-sampling based XORing [161] are applied.

Attack method	QID attribute combination	Window based XORing / Re-sampling based XORing 1-to-1	1-to-*many*	*many*-to-1	*many*-to-*many*	*wrong*
OptKro	FN and LN	0 / 0	0 / 0	0 / 0	0 / 1	0 / 0
	FN, LN, and SA	Experiments terminated due to high time consumption				
FrqChr	FN and LN	0 / 0	0 / 0	0 / 0	1 / 1	0 / 0
	FN, LN, and SA	Not enough frequency information found				
GrpMit	FN and LN	5,193 / 0	0 / 0	2 / 0	1,568 / 0	0 / 0
	FN, LN, and SA	4,690 / 0	0 / 0	0 / 0	16 / 0	0 / 0
ExFrqChr	FN and LN	0 / 0	0 / 0	0 / 2	2 / 0	0 / 2
	FN, LN, and SA	Not enough frequency information found				
PtMinVid	FN and LN	825 / 0	17 / 0	47 / 0	695 / 0	6 / 0
	FN, LN, and SA	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
GrpVid (BF)	FN and LN	12,747 / 12,133	0 / 12	0 / 0	4,218 / 4,062	0 / 3
	FN, LN, and SA	25,169 / 24,507	0 / 0	0 / 0	66 / 63	0 / 1

In Tables 9.10 and 9.11 we present the attribute and identity reidentification results for attacks applied on BFs that are hardened using window based XORing [161] and re-sampling based XORing [161] as we discuss in Appendix A. We only show the results for EURO (CS/CS) database pair as we are interested in learning whether these attack methods would work when BF hardening methods are applied. As can be observed, the **OptKro** attack method was not able to provide accurate attribute and identity reidentification results when hardening methods are applied. This is be-

Table 9.12: Time (in seconds) and memory (in Megabytes) consumption analysis of different attack methods when using different database pairs averaged over other parameter settings discussed in Section 9.3

Attack method	Time (sec) / Memory (MB)			
	EURO (CS/CIS)	NCVR (14e/14p)	MVR (16e/14p)	NCVR (21e/20p)
OptKro	2,311,492 / 3,232	–	–	–
FrqChr	1 / 4,566	5 / 5,363	5 / 5,298	547 / 30,433
GrpMit	33,096 / 6,089	76,382 / 11,775	45,489 / 11,033	–
ExFrqChr	57 / 4959	1,131 / 10,233	459 / 8,274	5,854 / 156,138
FrqVid	3 / 4,682	32 / 6,214	29 / 5,799	1,326 / 46,986
PtMinVid	8,777 / 12,549	17,594 / 50,800	24,890 / 19,475	–
GrpVid	6,108 / 28,074	17,082 / 51,635	16,812 / 22,842	–

cause the window based XORing and re-sampling based XORing hardening methods will alter the bit patterns of atoms generated by the DH which will lead to inaccurate assignment of q-grams to atoms. On the other hand, both the **FrqChr** and **ExFrqChr** attack methods worked (even though the results are low due to low frequency information) because they are based on aligning frequent QID values to frequent BFs, and those frequency information will not be altered by the two hardening methods.

Both the **GrpMit** and **FrqVid** attacks only worked for the window based XORing hardening method. Due to the use of a sliding window with a small window size ($w = 2$) in this hardening method, the resulting BFs can still retain some bit patterns as well as bit pattern frequencies which can be exploited by these attacks. However, it is worth noting that with larger window sizes (for instance $w = [50, 100]$) both the **GrpMit** and **FrqVid** attacks did not provide accurate results. Out of all the attacks the **GrpVid** attack obtained the highest reidentification accuracy. This is because the **GrpVid** attack does not depend on individual bit patterns, but on similarity neighbourhoods of encoded records, and both the hardening techniques we use do not alter the similarities calculated between records. Therefore, such hardening techniques are not practical in preventing similarity neighbourhood based attacks.

Finally, in Table 9.12 we show the runtime and memory consumption results for the different attack methods when using different database pairs. Note that these results are averaged over all the other parameter settings and encoded QID attribute combinations. As can be seen the **FrqChr** and **FrqVid** attack methods took the least amount of time and memory to run, requiring a minimum of 1.2 seconds and 2.5 seconds, respectively. With the large NCVR database pair these two attacks took around 9 and 22 minutes to complete, respectively. The most time consuming attack was the **OptKro** attack method requiring a maximum of around 27 days to complete. The second most time consuming attack was the **GrpMit** attack which took a maximum of around 21 hours to complete. Both the **PtMinVid** and **GrpVid** attacks had similar runtime and memory requirements overall across all the databases.

9.6 Discussion

According to results of the experimental evaluation we conducted to compare our proposed three attacks with four baseline methods, we identified both strengths and limitations associated with these attacks.

First, the **OptKro** attack only works when the BF encoding is conducted using the DH method [52]. Applying this attack on BFs encoded using any other hashing method (for instance, random hashing) will not yield accurate results. Even with the DH method, we were not able to obtain results for three attribute combinations with EURO database pairs, and two and three attribute combinations with all the other database pairs due to the very high time consumption in optimising the encoded atom and plaintext q-gram matrices. As we mentioned in the previous section we ran some experiments for over a month without achieving any results.

As we discussed in Chapter 4, the **GrpMit** attack only works when the adversary has access to all encoding parameter settings. Even with a single different parameter setting the attack has failed to provide accurate results in our experiments. Furthermore, if the QID values are not padded with two separate padding characters (at the beginning and end) the attack will not be able to obtain accurate attribute and identity reidentification results and it will also cause the attack to consume more time and memory because of the large number of potential paths. However, out of all the attack methods we experimented with (including our own attacks) the **GrpMit** attack had the best reidentification results as we showed in the previous chapter. This is because of its high accuracy in identifying the q-grams encoded in each BF. Nevertheless, the practicality of this attack is still a significant concern due to its requirement of the adversary to know all the used encoding parameter settings.

Both the **FrqChr** and **ExFrqChr** attacks provided the worst results out of all the attack methods. Even though these two attacks are fast and can be applied on very large databases, obtaining accurate reidentification results when two or more QID attributes are encoded is difficult. Furthermore, these attacks are based on frequency alignment of top t plaintext QID values with BFs in the encoded database where t is set by the adversary. In our experiments we set $t = [10, 20, 50, 100]$. The larger the range of t becomes the less accurate the results will be because of the potential wrong assignments between plaintext and encoded values.

Out of our proposed three attack methods the **FrqVid** attack obtained the worst results due to the lack of required frequency information in considered match-keys. However, unlike the other two attack methods the **FrqVid** attack is fast and therefore can be applied on very large databases. Interestingly, it can be observed that for databases with a small number of records, the attack methods (including the baselines) that exploit the frequency vulnerabilities of QID values and q-grams did not provide good reidentification results. This is because only a limited amount of frequency information is available in databases with a small number of records that can be exploited by frequency based attacks. In contrast, both the **GrpVid** and **GrpMit** attacks were still able to obtain good results even with smaller databases. This is because the vulnerabilities exploited by the **GrpVid** and **GrpMit** attacks are similarity

of QID values and co-occurrence of q-grams, respectively, which will not be affected by the sizes of the databases. One strong advantage of the **GrpVid** attack is that, unlike all the other attacks we experimented with, it can be applied on any encoding technique that calculates approximate similarities between encoded values.

We discuss the limitations of our proposed three attack methods and potential future research directions in detail in Chapter 10.

9.7 Recommendations for PPRL

We now provide a list of recommendations that should be followed when executing a real-world PPRL project in order to strengthen the privacy of the entities whose records are being linked. Apart from the chapter specific recommendations we discussed in Chapters 5 to 8, following these recommendations will also make the PPRL process more secure by making it more resilient to the privacy attacks we discussed in this thesis.

9.7.1 Technical Recommendations

Technical recommendations focus on providing directions for using suitable encoding techniques and parameters, and testing a sensitive database both before and after the encoding process for potential vulnerabilities. Our recommendations are:

1. Use encoding techniques and appropriate parameter settings that are not known to be vulnerable to any of the existing privacy attacks. Attribute-level BFs, for example, are vulnerable to frequency based attacks [42, 121]. Furthermore, the double hashing method and cryptographic long-term key encoding technique for BFs have also shown to be susceptible to certain privacy attacks. Therefore, when using BF encoding [177], we recommend to (1) use the random hashing method [180], (2) encode values from several QIDs into one BF, such as record-level BF encoding [61], and (3) apply BF hardening techniques that have shown to be resilient to existing attacks on PPRL [161].

2. Use salting to generate record specific encodings. Salting is the process of appending a record specific value such as date of birth to a string to create a different hash value [147]. Stable and reliable QID values such as year of birth and place of birth can be used as salts when hashing/encoding other QID values in records. This is an effective method to reduce the amount of frequency information that an adversary can gain from an encoded database. However, salting requires QID values that are complete, accurate, clean, and that do not change over time.

3. Use different encoding settings for different blocks (via salting) if the used blocking method generates non-overlapping blocks. This can be a strong defence mechanism against existing attacks on PPRL because the generated encodings in distinct blocks will be different from one another, and therefore the

amount of frequency information that can be extracted will be reduced. However, this method would not work if the blocks are overlapping since the same record will have different encodings depending on the blocks that a record has been inserted into.

4. Assess the trade-off between privacy, linkage quality, and scalability based on the specific requirements of a linkage project. If the linkage project requires minimal scalability with improved privacy we recommend to use secure multi-party computation (SMC) based techniques, as we discussed in Section 2.5, for PPRL because those techniques are provably secure and no attacks have so far been proposed specific to SMC based techniques.

5. Test for the vulnerabilities discussed in Chapter 5 (and in Section 3.6.3) in both the plaintext and encoded sensitive databases. Understanding how vulnerable certain QID values in a sensitive database are will allow the database owners (DOs) to make necessary decisions related to selecting a suitable encoding technique, only encoding not vulnerable QID values, and potentially removing or perturbing certain vulnerable values.

6. Test for the overlap between the sensitive database and databases that are publicly available. As we discussed in Section 3.5.3 and illustrated in Figure 3.3, both row-wise and column-wise overlaps can be measured. The column-wise overlap can be restricted by selecting QID values that are less likely to be publicly available yet which provide good linkage quality. Row-wise overlap, on the other hand, cannot be directly eliminated without removing the overlapping records and doing so might lead to lower linkage quality.

7. Run existing privacy attacks on the encoded database using publicly available databases to test if it can be successfully attacked. This is a practical approach to investigate whether there is enough information in publicly available databases to attack a sensitive database that is encoded using a certain encoding method and a given set of parameters. Furthermore, such an assessment will also help decide if an attack is feasible in terms of time and memory requirements. If an attack is possible then the DOs can change the used encoding technique accordingly.

8. Perform parameter tuning based on the selected QID values of the databases to be encoded and linked. As opposed to using any recommended parameter settings, a careful selection of parameter values, which will provide good linkage quality as well as privacy protection, will potentially limit the adversary's ability to guess certain parameter values that have been used for a linkage. This is because unlike the recommended parameter settings, the tuned parameter values for individual databases are not publicly available, and the adversary might not be able to gain access to those parameter values. However, it is worth noting that some parameter settings used in certain PPRL techniques,

such as q-gram length and the hashing method for BF encoding, do only have a small number of suitable options [40, 75].

9. Accrediting of computer systems, software, and algorithms used for PPRL projects will ensure the security of these system by evaluating them with regard to different security standards. Note that accreditation will not guarantee the security or privacy of the sensitive data itself rather it will only provide a certification for the computer system, software, and algorithms used.

9.7.2 Non-Technical Recommendations

The following recommendations focus on non-technical factors that will influence the security and privacy of the sensitive data used in a PPRL project. Our recommendations are:

1. Provide proper training for the parties and their employees involved in a PPRL project. Suitable training programs will instruct both employers and employees to effectively recognise and prevent social engineering attacks. Such training should not be a one-time undertaking, but a continuous process where employees are regularly trained on how and when to protect the sensitive data that they are entrusted with.

2. Encourage all the parties involved in a PPRL project to follow best practice approaches at all times. These best practices include implementing organisational structures to protect sensitive information, such as incident response plans and following the Five Safes framework for risk assessment in data access [50], implement data access privileges to different parties, develop data privacy and security policies within the organisation, always track and monitor the usage of sensitive data, use strong passwords and secret keys that cannot be easily guessed, and so on.

3. Follow all required ethical and legal standards such as the European Union General Data Protection Regulation (GDPR), the United States' Health Insurance Portability and Accountability Act (HIPAA)[4], or the Australian Privacy Principles (APPs)[5], as put forth by governments and other organisations to protect sensitive information and ensure the privacy of entities.

While the combination of these technical and non-technical recommendations will make a PPRL project more secure, given the lack of provable privacy guarantees of many existing PPRL techniques (as discussed in Section 2.5) [40], further research into novel and more secure PPRL techniques is still required.

[4]https://www.cdc.gov/phlp/publications/topic/hipaa.html
[5]https://www.oaic.gov.au/privacy/australian-privacy-principles

9.8 Summary

In this chapter we conducted a broad experimental evaluation of our proposed three attack methods and compared them with four state-of-the-art attack methods discussed in Chapter 4. We evaluated all attack methods using three separate databases and different variations of them containing different number of records and error characteristics. The experimental results on these databases showed that our proposed attacks are able to obtain high reidentification accuracy using less knowledge of the encoding parameter settings compared to the state-of-the-art attack methods. Furthermore, some state-of-the-art attack methods require high memory and time consumption despite being able to provide accurate reidentification results. One of our proposed attack methods, on the other hand, can be applied on very large real-world databases, while another is capable of attacking any PPRL encoding technique that calculates approximate similarities between records.

Our proposed attack methods can further be evaluated using other real-world databases with different characteristics and different numbers of records to understand their applicability in real-world linkage situations. One potential direction is to evaluate the performance of our attacks in a situation where encoded database is from one state (or country) and the plaintext database used by the adversary to attack the encoded database is from another state (or country) but with same QID attributes in both the databases. The evaluation conducted in this chapter will serve as a foundation for such further work providing necessary guidelines. In the following chapter we conclude the thesis by summarising our work and discussing potential future research directions.

Chapter 10

Conclusion and Future Work

In this thesis we have conducted a research study into identifying and analysing the vulnerabilities and limitations of existing privacy-preserving record linkage (PPRL) techniques. In the previous chapters we have proposed a taxonomy of attacks on PPRL, a framework to assess the vulnerabilities of sensitive databases used in PPRL, and three novel attack methods that can be applied on different PPRL techniques. In this chapter we summarise our contributions and discuss the conclusions we can make based on our findings.

In Section 10.1 we first briefly outline the identified challenges of our research. In Section 10.2 we then summarise the contributions we made to address those challenges and achieve the corresponding aims and objectives. In Section 10.3 we next discuss the current limitations associated with our work, and in Section 10.4 we describe potential directions for future research in the area of attacks on PPRL. Finally, in Section 10.5 we conclude our thesis.

10.1 Outline of the Research Challenges

As we described in Chapter 2, PPRL focuses on developing techniques that will allow the linkage of records using personally identifiable information of entities without compromising the privacy of those entities [176, 200]. Even though a number of different PPRL techniques have been proposed over the years, recent research has shown that some of the popular PPRL techniques are susceptible to privacy attacks [119, 121, 139, 147]. However, these existing attack methods have various limitations, including impractical assumptions about the adversary's prior knowledge on encoding parameters, the ad-hoc environment that the attacks are being applied in, and excessive runtime and memory requirements. Furthermore, no systematic analysis has been carried out to characterise and compare existing attack methods on PPRL.

Based on the shortcomings and gaps of existing work, we have identified the main challenges of our research work: (1) how do we analyse and categorise existing attack methods on PPRL based on their characteristics, (2) how do we quantify the vulnerabilities that exist in both encoded and plaintext databases that contain sensitive information, and (3) how do we exploit the weaknesses of popular and recently

proposed PPRL techniques using novel attack methods that are more practical, accurate, and efficient. In the following section we describe how we addressed these identified challenges by achieving the aims and objectives of our research (as detailed in Chapter 1) through our contributions.

10.2 Summary of the Contributions

We have made five contributions to meet the aims and objectives of our research as we detailed in Chapter 1.

1. **A taxonomy of existing attack methods on PPRL**: In Chapter 3 we have proposed a novel taxonomy of attacks on PPRL where we have identified twelve dimensions that can be used to characterise these attack methods. In our taxonomy we have classified the twelve dimensions into three aspects: adversary, technical, and practical. Under adversarial aspects we discussed how to characterise an adversary who is conducting an attack using the dimensions types of adversaries, adversary's prior knowledge about the encoding parameters and accessibility to data, and non-technical issues of an attack. Under technical aspects we discussed the technical characteristics of an attack including the types of attack methods, the exploited vulnerabilities in encoded databases, the attack scope, and the computational complexity of an attack. Finally, under practical aspects we described the empirical dimensions of an attack, which includes measuring the accuracy and scalability of an attack, the implementation characteristics of an attack, and the databases used for the experimental evaluation of an attack.

   To the best of our knowledge, no such taxonomy currently exists in the context of attacks on PPRL. The owners of sensitive databases can utilise our taxonomy to evaluate and compare different attack methods on PPRL in order to identify their strengths and limitations, and to assess whether those attacks will be successful on their own sensitive databases. By proposing this taxonomy we have addressed the objective of conducting a broad analysis of attacks on PPRL as specified in Chapter 1.

2. **A vulnerability assessment framework for PPRL**: In Chapter 5 we have proposed a novel framework for assessing the vulnerabilities associated with both plaintext and encoded databases as used in the context of PPRL. We have identified five characteristics of such databases: frequency, length, co-occurrence, similarity, and similarity neighbourhood. These can be exploited by an adversary in an attack to reidentify encoded quasi-identifying (QID) values. As we discussed in Chapter 5, all existing attack methods exploit one or more of these vulnerabilities of databases encoded using different PPRL techniques.

   In an experimental evaluation we used our framework to calculate the vulnerabilities of a real-world and a synthetic database encoded using five PPRL

techniques. Given the increasing demand for an effective method to assess the vulnerabilities of the databases used in PPRL and for secure PPRL encoding techniques, such a systematic framework can be highly beneficial. By proposing this framework we have addressed the objective of measuring the vulnerabilities of sensitive data and PPRL encoding techniques.

3. **A frequency based attack on multiple match-key encoding**: In Chapter 6 we have presented a novel privacy attack on a recently proposed PPRL technique named multiple match-key (MMK) encoding [163]. The proposed attack employs a set of statistical tests to calculate the correlation between the frequency distributions of encoded match-keys and plaintext QID value combinations. By analysing these correlations the attack identifies the plaintext attribute combinations used to generate encoded match-key values. The highly correlated plaintext QID combination and encoded match-key pairs are aligned with each other, and these alignments are then used to reidentify plaintext QID values encoded in individual match-key values.

   The experimental evaluation we conducted showed that our attack can successfully reidentify encoded plaintext QID values, and therefore MMK encoding is susceptible to such frequency based attacks. We have also proposed two countermeasures to improve the privacy of the MMK encoding technique and have discussed the trade-offs between linkage quality and improved privacy when these countermeasures are applied.

4. **A pattern-mining based attack on Bloom filter encoding**: In Chapter 7 we have proposed an extension of a cryptanalysis attack on Bloom filter (BF) encoding [177] initially proposed by Christen et al. [43]. The attack employs a maximal frequent pattern-mining algorithm to identify sets of frequent bit positions in a BF database that can encode frequent q-grams. In our extended version of the attack, we propose a probabilistic language model and a partitioning based frequency analysis to expand the identified sets of q-grams and their corresponding bit positions. These identified q-grams and their bit positions are then used to reidentify plaintext QID values encoded in each BF.

   The experimental evaluation conducted on this attack showed that our proposed extension can outperform the initial version of the attack [43] in terms of the number of accurate reidentifications. Our attack was able to successfully reidentify encoded values when more than one QID value is encoded into a single BF (making all BFs in an encoded database unique), a situation where most other attacks on BF encoding would fail. This makes our attack feasible in real-world scenarios compared to other attacks. Finally, we have described a set of potential countermeasures to prevent this types of attacks and their impact of linkage quality over improved privacy.

5. **A graph matching based attack on PPRL**: The two attack methods we proposed in Chapters 6 and 7 can only be applied on one specific PPRL encoding

technique, namely BFs. In Chapter 8 we have proposed a novel attack based on graph matching which, unlike earlier attack methods, can be applied on a database encoded using any PPRL technique that calculates approximate similarities between values. The initial step of the attack is to generate two similarity graphs from the plaintext and the encoded databases based on the similarities calculated between values in each database. For each node in both graphs the attack then calculates a set of features based on the neighbourhood of that node. This results in one feature matrix for the encoded and one for the plaintext graph, where each node will have a feature vector assigned to it. The attack next compares feature vectors of nodes across the two graphs to measure the similarities between these nodes. Based on the calculated similarities the attack finally aligns plaintext nodes (QID values) with encoded nodes (encodings).

In an experimental evaluation we applied our attack method on three PPRL techniques, BF encoding [177], tabulation min-hash (TMH) encoding [186], and two-step hash (2SH) encoding [158]. The results of this evaluation showed that with certain parameter settings our attack can successfully reidentify a large number of encoded QID values with all three encoding techniques. This highlights the vulnerabilities of more recently proposed techniques such as TMH encoding [186] and 2SH encoding [158], and therefore the need for further research into more secure PPRL encoding techniques.

We have addressed the last three objective of our research, as described in Chapter 1, by proposing the above three attack methods for PPRL. Unlike existing attacks our proposed attack methods require less amount of knowledge about encoding parameters and they are more accurate in reidentifying encoded attribute values. Our attack on MMK encoding can be applied to large encoded databases with millions of records, as we illustrated in Chapter 9. We also evaluated our attacks using different parameter combinations and three databases to evaluate their applicability in various real-world PPRL settings.

10.3 Limitations of the Research

In the following we discuss the current limitations of the work proposed in this thesis.

- The taxonomy of attacks on PPRL we proposed is a conceptual analysis of existing attacks based on their characteristics. All existing attacks on PPRL are, however, proposed for perturbation based PPRL techniques as we discussed in Chapter 4. To the best of our knowledge no attack has so far been proposed for secure multi-party computation (SMC) based PPRL techniques. Therefore, it is worth noting that an attack method on a SMC based technique might have different characteristics other than the ones we discussed in our taxonomy.

- According to the experimental evaluation results of the frequency based attack on MMK encoding (as discussed in Chapter 6), it can be concluded that this

attack performs better when the size of the encoded database increases and the number of QIDs in match-keys decreases. This is because with larger databases and a smaller number of QIDs in match-keys the frequency distributions of the match-keys will be more distinct, and therefore they can be uniquely identified and exploited by the attack successfully.

- The success of the pattern-mining based attack on BF encoding [177] (as discussed in Chapter 7) depends on the accuracy of the first step of the attack which is applying maximal frequent pattern-mining to identify the bit positions that frequent q-grams are hashed into. Both the second and third steps of this attack are based on these identified frequent q-grams and their corresponding bit positions. If the first step of the attack is not successful in identifying bit positions of frequent q-grams accurately, then the second and the third steps will also have inaccurate results which will lead to the wrong reidentification of plaintext QID values in the fourth step. However, it is also worth noting that according to our experimental results presented in Chapter 7, for most parameter settings the attack provided highly accurate results for the first step.
- There are two main limitations to our proposed graph matching attack for PPRL (as discussed in Chapter 8). First, if the differences between the encoded and plaintext similarities are high and if the relationship between these similarities is not linear, then it will be difficult to obtain accurate reidentification results. This can be seen in our experimental evaluation of the attack on different databases with the different number of matching records, with 2SH encoding, and without similarity adjustments. Second, the more attributes are encoded the lower the accuracy of reidentifications. This is because of the differences in similarity values across the two graphs, as well as having similar graph neighbourhoods for dissimilar nodes, as we discussed in Section 8.6.

10.4 Future Research Directions

In this section we discuss several open research avenues that could be explored in terms of both our contributions as well as attacks on PPRL in general. These directions for further research unfold from our observations and conclusions on analysing various PPRL encoding techniques using our proposed taxonomy, vulnerability framework, and attack methods.

- **Improvements to the attack methods**: The three attack methods we proposed in Chapters 6 to 8 can potentially be further improved as follows.
  - Investigate possible filtering techniques for match-keys in the frequency based attack on MMK encoding to improve the performance and accuracy of the attack. One such technique could be to analyse if multiple match-keys are the same for a given record pair. Such patterns of the same match-keys in record pairs can potentially lead to the identification

of QIDs used in encoded match-keys and thereby the reidentification of encoded plaintext values.

  - Explore different pruning techniques on the main algorithms of the pattern-mining based attack on BF encoding to increase its efficiency. Such pruning techniques in the first three steps of the attack will reduce the number of pattern-mining iterations that the attack would perform and therefore will make the attack more efficient.
  - Potential future research directions for the graph matching based attack on PPRL could be to: (1) investigate supervised and unsupervised machine learning methods to conduct the alignment between encoded and plaintext similarities without having knowledge of the encoding parameters used, and (2) explore the effectiveness of the different node features (as discussed in Chapter 8) in identifying a node uniquely based on its neighbourhood and analyse whether these features have a trade-off of efficiency for improved accuracy.
  - Apart from the above mentioned future research directions for the individual attack methods, all three attack methods presented in this thesis can be parallelised using distributed computing frameworks such as MapReduce [49]. Parallelisation of attacks can improve their scalability and make these attacks more applicable on very large real-world databases.

- **Attacks on numerical PPRL encoding techniques**: Our proposed attack methods in Chapters 6 to 8 are evaluated on MMK, BF, TMH, and 2SH encoding techniques. While MMK encoding can also be used for numerical QID values such as age, date of birth, and height, encoding techniques such as BF, TMH, and 2SH are mainly used for encoding string values such as names and addresses. As we discussed in Chapter 2, only a small number of encoding techniques have been proposed for numerical values [106, 205] and no attack have yet been proposed for any of those numerical encoding techniques. Therefore, it is worth investigating possible attack methods for numerical encoding techniques to explore their limitations and vulnerabilities.

- **Efficient SMC based PPRL techniques**: The research presented in this thesis is mainly focused on perturbation based PPRL techniques as they are more popular in practical applications compared to the SMC based techniques because of the ease of implementation and their scalability. As we discussed in Chapter 2, while SMC based techniques have high complexity, they also guarantee provable privacy. On the other hand, computing resources and power increase every day making more resource-intensive applications feasible. Therefore, we believe it is worth investigating the development of SMC based techniques and making them more efficient in terms of their complexity. The use of SMC based techniques in real-world PPRL projects will be highly beneficial because of their provable privacy guarantees.

- **Attacks on SMC based PPRL techniques**: The attack methods proposed in this thesis can only be applied on perturbation based PPRL techniques. Even though the graph matching attack can theoretically be applied on a SMC based techniques that calculates approximate similarities between records in an encoded database, we have not analysed this situation in our experimental evaluation of the attack. Therefore, it is important to explore possible attacking scenarios for SMC based PPRL techniques in order to evaluate the privacy guarantees of those techniques.

10.5 Conclusion

The aims of this thesis were to *quantify the vulnerabilities that exist in PPRL encoding techniques, explore novel attack methods for PPRL techniques to identify the limitations of these techniques, and evaluate the developed attack methods in terms of their reidentification accuracy*. We achieved these aims by, (1) proposing a taxonomy of attacks on PPRL to analyse the characteristics of existing attacks, (2) developing a framework for PPRL to quantify the vulnerabilities in databases encoded using different PPRL techniques, (3) developing three attack methods on different PPRL encoding techniques to demonstrate their limitations with regard to preserving the privacy of the entities, and (4) evaluating the proposed attack methods using various parameter settings and databases to analyse their applicability in real-world scenarios.

On one hand, our research highlights the need for more secure PPRL encoding techniques while on the other hand it provides a set of tools for database owners, evaluators, and researchers to analyse the privacy guarantees of any existing PPRL encoding technique. Further, we would like to emphasise that the intention behind this thesis is not to encourage malicious adversaries with potential attack ideas. The focus is on discouraging them by revealing the vulnerabilities of certain PPRL encoding techniques and parameter combinations, so that better PPRL techniques can be developed and database owners can take actions to prevent such vulnerabilities from being exploited.

Appendix A

Bloom Filter Encoding for Privacy-Preserving Record Linkage

In this appendix we describe different aspects of Bloom filter (BF) encoding for privacy-preserving record linkage (PPRL) in detail. In Appendix A.1 we describe the BF encoding process, false positive probability of BFs, and optimal parameter settings for the encoding of BFs. In Appendices A.2 and A.3 we discuss different hashing and encoding techniques, respectively, used in BF encoding for PPRL. Finally, in Appendix A.4 we provide details on a few selected hardening techniques proposed for BFs to strengthen their privacy guarantees.

A.1 Bloom Filter (BF) Encoding

Bloom filters (BFs) were proposed by Burton H. Bloom [24] in 1970 for representing a set, **s**, of elements and for querying set membership functions efficiently. A BF is a space-efficient probabilistic data structure, generally defined as a bit array with a specified length l. Each element of a set, $s \in \mathbf{s}$, is hashed into this bit array using a set of hash functions. When an element, s', is queried from the BF to check for the membership, the result of the query will be either s' is *possibly in set* **s** or *definitely not in set* **s**. Due to the possibility of hash collisions false positives might occur where an element is recognised as in the set while in fact it is not. However, false negatives are not possible as long as the elements hashed into the BF are not removed after the hashing process. With more elements being hashed into a BF, the probability of false positives generally increases [140].

Employing BFs for PPRL was proposed by Schnell et al. [177] in 2009. The similarities of pairs of BFs were measured using Dice coefficient similarity [38]. Since BF encoding was first proposed, it had become popular among researchers and many PPRL application domains because of its efficiency and easy implementation [10, 29, 153, 154, 164].

A.1.1 Bloom Filter Encoding Process

Given a BF $\mathbf{b}$ of length l bits and a set of elements $\mathbf{s}$, the encoding of each element $s \in \mathbf{s}$ into $\mathbf{b}$ is performed as follows. Initially all bits in $\mathbf{b}$ are set to 0. Then each element $s \in \mathbf{s}$ is hashed into $\mathbf{b}$ using $k \geq 1$r independent hash functions, h_i, $1 \leq i \leq k$. Each hash function returns a bit position within the range $[1, \ldots, l]$ for each element in $s \in \mathbf{s}$ and those bit positions will be set to 1 in $\mathbf{b}$ i.e. $\mathbf{b}[h_i(s)] = 1$, $1 \leq i \leq k$.

When querying an element s' for the set $\mathbf{s}$, we use the same set of hash functions h_i, $1 \leq i \leq k$, to get the set of k bit positions for the element s'. If any of the bit positions is 0, i.e. $\exists\ 1 \leq i \leq k : \mathbf{b}[h_i(s')] = 0$, then the element s' is definitely not in the set $\mathbf{s}$. If all bit positions are set to 1, i.e. $\forall\ 1 \leq i \leq k : \mathbf{b}[h_i(s')] = 1$, then the element s' is likely in the set $\mathbf{s}$. However, because of hash collisions [51], one or more bit positions set by s' can also be set by other elements in the set $\mathbf{s}$ resulting in a false positive.

In the context of PPRL the elements of the set are the q-grams extracted from one or more QID values in a record. The generated BFs can then be used to calculate approximate similarities between records, as required by the PPRL process, using a set based similarity function such as the Dice coefficient similarity [177].

In a PPRL protocol each database owner (DO) encodes her QID values into BFs and then these BFs are either sent to a linkage unit (LU) to calculate the similarities between BFs or the similarities are calculated by the DOs themselves [124, 202]. The latter can be achieved using secure multi-party computation based techniques, as we discussed in Chapter 2. After calculating the similarities the compared BF pairs are classified into matches, non-matches, or alternatively potential matches [177].

A.1.2 Probability of False Positives

As discussed above, with BFs there is a possibility of getting false positives. Hence it is vital to understand the probability of getting a false positive and how to select optimal parameter settings in order to balance the aspects of linkage quality, scalability, and privacy in PPRL applications.

For a given BF length $l \geq 1$, number of hash functions $k \geq 1$, and number of q-grams $m \geq 1$, which are hashed into the BF, assuming that each hash function is uniform and independent from all others, the false positive probability ρ_f can be calculated as [28, 31],

$$\rho_f = \left(1 - \left[1 - \frac{1}{l}\right]^{km}\right)^k \approx \left(1 - e^{-km/l}\right)^k. \tag{A.1}$$

As can be seen, the false positive probability ρ_f decreases when the BF length l increases, and increases when the number of q-grams m increases.

A.1.3 Optimal Parameter Setting

Even though the false positive probability increases with the number of q-grams, m, m cannot be defined by the user since it depends on the lengths of QID values that need to be encoded. However for a given number of q-grams, m, and a defined BF length, l, the optimal number of hash functions, k_{opt}, can be derived to minimise the number of hash collisions in BFs, and by extension potentially minimise the false positive probability. k_{opt} is calculated by getting the derivative of Equation A.1 and minimising it with respect to the number of hash functions k [140], resulting following equation:

$$k_{opt} = \frac{l}{m} ln(2). \tag{A.2}$$

For a given error tolerance rate (false positive probability), ρ_f, by applying k_{opt} into Equation A.1, we can calculate the length of the BF, l, as,

$$l = -\frac{m\ ln\ \rho_f}{(ln2)^2}. \tag{A.3}$$

As shown in [140], using Equation A.1, the false positive probability can be minimised when each bit of the BF is set to 0 with the probability of 0.5 i.e. by optimising k for a given l and m [27].

The value for m can be estimated by analysing the QID values in the databases that are to be encoded. By calculating the number of q-grams for each QID value we can get the average number of q-grams per encoded value in a database. This value, along with the BF length is used to compute the optimal number of hash functions, k_{opt}, which can then be used be used for encoding of QID values. With increasing length of BF, l, the value of k_{opt} will also increase. Larger k_{opt} will have more memory and runtime requirements.

While l and k_{opt} determine the computational complexity of a PPRL process, the false positive probability, ρ_f, determines the linkage quality and privacy of the PPRL. With a higher value of ρ_f there will be more false positives which will lead to a reduction of linkage quality. On the other hand a higher ρ_f value will provide improved privacy since an adversary would not be able to guess the QID values encoded into each BF with high accuracy due to the large number of collisions [64, 177, 201]. Hence it is important to note that with BF encoding for PPRL there is always a trade-off between linkage quality and privacy.

Selecting optimal parameter settings for BF encoding was recently explored further by Borgs [27] where a model based solution was proposed. Borgs evaluated the usability of four models, a simple linear model, a Bayesian information criterion model, a response surface model, and a random forest based regression model, to estimate the optimal parameter choices for BF encoding. The results of this evaluation showed that the model based solutions provided better results in terms of linkage quality as well as privacy compared to using k_{opt} to minimise the collisions [27].

A.2 Bloom Filter Hashing Techniques

As we discussed in Appendix A.1.1, BF encoding requires a set of hash functions to hash the q-grams of QID values into BFs. Over the years four hashing techniques have been proposed for BF encoding to improve the efficiency of the hashing process. In this section we discuss two of those techniques as we conducted the experimental evaluations of our proposed attack methods using these two techniques. These hashing techniques use different equations to convert a hash value of each q-gram into a BF bit position index value.

A.2.1 Double Hashing (DH)

When encoding an element into a BF a set of independent hash functions are usually applied to each individual element in the set. But this process has a significant performance overhead due to the use of multiple independent hash functions (k). When the number of elements, m, and the number of hash functions, k, grow the required computational complexity to encode these elements into a BF increases quadratically [52]. In order to address this issue of performance, the double hashing (DH) method was proposed by Dillinger and Manolios [52] in 2004.

The idea behind DH is to use only two hash functions, instead of a sequence of k hash functions, and apply simple arithmetic operations to calculate all required BF indices for each element that need to be encoded. Given an element s, two independent hash functions $h_1()$ and $h_2()$, and the required number of BF indices k, the DH scheme can be applied to calculate each BF bit position $g_i(s)$, with $1 \leq i \leq k$, that need to be set to one, as follows [177]:

$$g_i(s) = 1 + [h_1(s) + i \cdot h_2(s)] \; mod \; l. \tag{A.4}$$

First the element s is hashed using two hash functions $h_1()$ and $h_2()$. The resulting two hash values are then summed together after multiplying $h_2(s)$ where i represents the ith index. The modulo operation is performed in order to make sure the index value we obtain is a valid index which falls between the range $[1,\ l]$.

The DH mechanism was empirically analysed by Dillinger and Manolios [51, 52] and later theoretically analysed by Kirsch and Mitzenmacher [112] to show it significantly reduces the runtime while maintaining almost similar accuracy compared to naive independent hashing. Even though it has been shown that the DH consumes less memory and runtime, it also contains a weakness which makes BF encoding with DH vulnerable to frequency attacks. Compared to the number of all possible bit patterns that a BF can get using a certain number of hash functions, the DH is only able to generate a much smaller number of bit patterns. As we discussed in Chapter 4, this property of DH has led to the development of frequency based cryptanalysis attacks [119, 147], on BFs.

A.2.2 Random Hashing (RH)

The DH technique discussed above has a limited output space which is smaller when compared to global output space. Therefore, DH is prone to bit pattern based attacks on BFs as we discussed in Chapter 4. To prevent such attacks on PPRL, the random hashing (RH) was proposed by Schnell and Borgs [180] to replace the DH scheme. The implementation of DH was inspired by the concept of using random hash functions, to make BFs more resilient against privacy attacks. The idea behind the RH scheme is to use the encoding element, s, as a seed to a pseudo-random number generator (PRNG) in order to generate $k \geq 1$ random BF indices. These k indices are then used to set relevant bit positions in the the BF to 1.

For a given element, s, and a PRNG, pg, the k BF bit positions $g_i(x)$, with $1 \leq i \leq k$, to be set to 1 can be calculated as follows:

$$\begin{aligned} & pg.seed(s), \\ g_i(s) = & pg.randint(1,\ l), \end{aligned} \tag{A.5}$$

where we assume that the $pg.seed()$ function sets the seed specified to the element, s, and the $pg.randint(s,\ e)$ function outputs a randomly selected integer from the range $[1,\ l]$. For a specific seed s, the PRNG will always generate the same sequence of pseudo-random numbers, which makes the similarity calculation between BFs possible. Here, instead of using the element s as the seed of PRNG directly, we can use a single hash function, $h_1()$, to hash the element s and used the hash value $h_1(s)$ as the PRNG seed, which makes the RH scheme more secure since an adversary then also needs to be able to guess the specific hash function used in order to break the encoding process.

Unlike the DH approach, the RH scheme will not generate any repeatable patterns because it is not based on the linear combination and addition of two integer hash values. This makes the RH approach more difficult to attack compared to the DH approach. Nevertheless, as we discussed in Chapter 4, the RH technique is also susceptible to certain frequency based attack methods [42].

A.3 Bloom Filter Encoding Techniques

Using the hashing techniques discussed in Appendix A.2, the encoding of QID values into BFs can be conducted in several ways. In this section we describe three encoding techniques proposed for BF encoding over the years.

A.3.1 Attribute-Level Bloom Filters (ABF)

Attribute-level BF (ABF) encoding, also known as field-level BF (FBF) encoding, was initially introduced by Schnell et al. [177] in 2009. ABF encodes a single QID value to a BF which results in multiple BFs for a single record if multiple QID values are used in the linkage process.

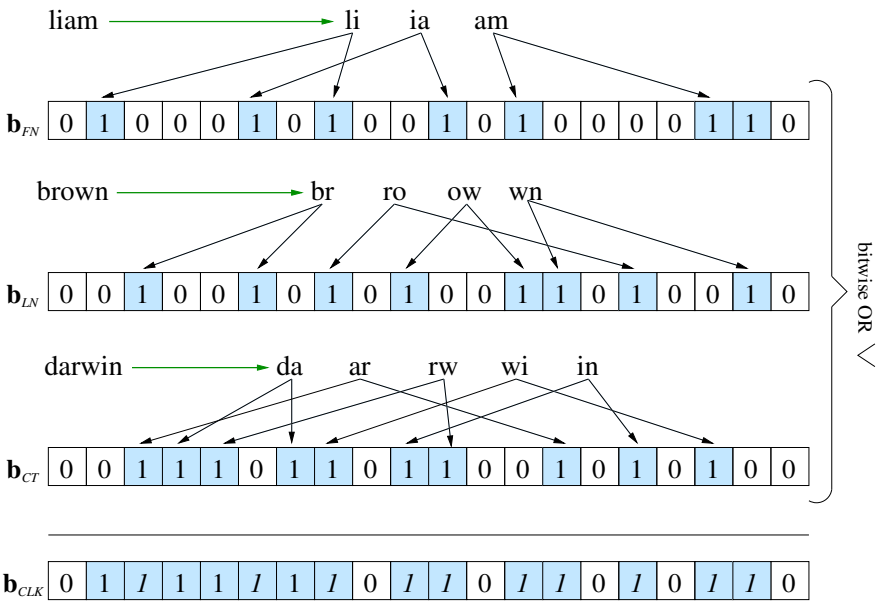


Figure A.1: An example illustration of the cryptographic long-term key (CLK) BF encoding technique (adapted from [40]) using values from three QID attributes *First-Name* (liam), *LastName* (brown), and *City* (darwin) where $l_q = 2$, $k = 2$, and $l = 20$ parameter settings are used for the encoding. Collisions are illustrated in italic font.

In ABF encoding, first each QID value is converted into a set of q-grams. Before converting to the q-grams, the QID value may or may not be padded with a special character at the beginning and end of the value. For instance, if the QID value 'peter' need to be padded and converted into bigrams ($l_q = 2$), them the both ends of the string must be padded with a non alphanumerical character such as '_'. This will result the value '_peter_', and when converted into bigrams we get the set {_p, pe, et, te, er, r_}. It is also important to note that with q-grams of length l_q, $l_q - 1$ non alphanumerical characters need to be padded at each end of the QID value. Generally padding of string values are considered because, (1) the beginning and end of the QID values being encoded need to be emphasised then padding should be used, and (2) every character of the QID value will end up in l_q number of q-grams, making the character distribution in the set of q-grams consistent.

Once we get the set of q-grams from the QID value a BF can be generated by hashing each q-gram into the BF using any of the hashing techniques discussed in Appendix A.2. If multiple QID values from a single record need to be encoded, each QID value is encoded in a separate BF, resulting a set of BFs for each record.

As shown by Schnell et al. [177], ABFs can provide highly accurate linkage results. However, as we discuss in Chapter 4, the basic ABF encoding is prone to frequency based cryptanalysis attacks [119, 121, 147]. These attacks exploit the frequency information of QID values in order to align them with frequent BF bit patterns.

A.3.2 Cryptographic Long-term Key (CLK)

The Cryptographic long-term key (CLK) encoding technique for BFs was proposed by Schnell et al. [178] to enable error tolerant comparisons between records when the records with typographical errors in QID values are considered. Apart from withstanding errors in records, CLK also provides improved privacy for sensitive values that are being encoded, when compared to ABF encoding. The basic idea behind CLK is to encode multiple QID values from a single record into one BF, resulting a single BF for each record. In CLK, all QID values are encoded using the same hashing method and the same number of hash functions, k. [178].

The basic steps of the CLK method are illustrated in Figure A.1. As can be seen, three QIDs *FirstName*, *LastName*, and *City* with values 'john', 'smith', and 'sydney', from a single record is required to be encoded into a single BF. A q-gram set from each QID value is extracted initially resulting in three sets of q-grams. As with ABF, QID values can be padded based on the requirements. Conceptually the process of hashing several q-gram sets can be viewed as first hashing each q-gram set into a separate ABF and then combining those ABFs into a single BF using the bit-wise OR operation. However, this process can be executed in a more memory efficient manner by combining the sets of q-grams from QID values into one set, and hashing the combined set of q-grams into one BF. This will improve the performance of the encoding process.

Since CLK encodes several QID values into a single BF, it can potentially result in reduced linkage quality compared to ABF encoding. This is because CLK allows only one similarity calculation per record in contrast to ABF's multiple similarity calculations and there can potentially be more hash collisions in CLK which can lead to false positives. However, according to the experiments conducted by Schnell et al. [178] on several databases, CLK was capable of achieving almost similar linkage quality compared to naive plaintext comparisons. Alternatively, CLK encoding will provide improved privacy compared to ABF since less frequency information is available to an adversary. Nevertheless, our pattern-mining based attack and graph matching based attack described in Chapters 7 and 8 have shown that CLK encoding is vulnerable to certain attacks even when all BFs in a database are unique.

A.3.3 Record-Level Bloom Filters (RBF)

As discussed in Appendix A.3.1 ABFs are vulnerable to frequency based cryptanalysis attacks [41, 42]. By utilising a simple frequency analysis of QID values Durham et al. [61] have shown that ABFs can be attacked using their bit frequency distributions. To overcome this security flow, Durham et al. [61, 64] have proposed record-level BF (RBF) encoding for BFs. Similar to CLK, RBF generates a single BF encoding for several QID values from a record, but it employs different techniques to do so.

Unlike CLK, RBF initially generates one ABF per QID value of a record. The lengths of ABFs can be different based on the average number of q-grams of each QID value. RBF randomly samples a set of bits from each ABF and then combines

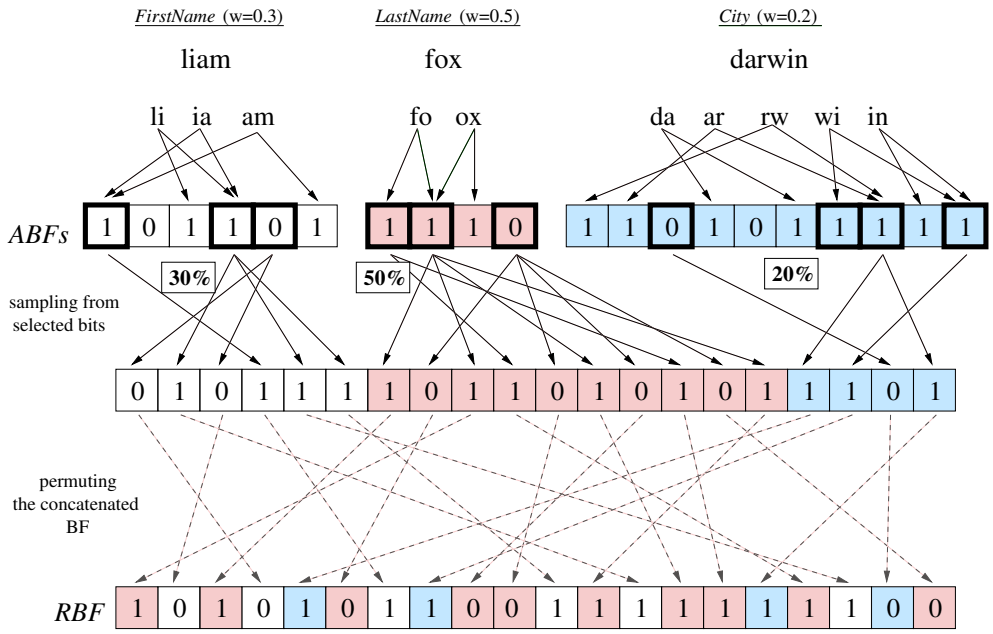


Figure A.2: An example illustration of record-level BF (RBF) generation (adapted from [61]). First, the three QIDs *FirstName*, *LastName*, and *City* with values 'liam', 'fox' and 'darwin' have been encoded into separate QID-level BFs (ABFs) of lengths $l^{FN} = 6$, $l^{LN} = 4$, and $l^{CT} = 10$, and the same number of hash functions $k = 2$. Then certain bit positions (identified in bold outline) are selected. The next step is to sample a certain number of bits from the selected bit positions using the respective weights $w_1 = 0.3$, $w_2 = 0.5$, and $w_3 = 0.2$ for each QID, and to concatenate these sampled bit positions into one BF for the record. In the final step this concatenated BF is permuted to get the final RBF of length $l = 20$.

these bits to generate a single BF. This mechanism was shown to be more secure than ABF and CLK [40]. The generation of a RBF consists of four steps.

1. **ABF generation**: For each QID attribute value in a record a separate ABF is first generated using a given k number of hash functions. where the length l of these ABFs can either be static or dynamic. When the length is set to static, a constant value for l is used for all ABFs of different QID values. On the other hand when the length is set to dynamic, the length of each ABF is calculated based on the probability, ρ, that a bit remains unset given the average number of q-grams occur in the corresponding QID attribute. For instance, assume the average number of q-grams in QID attribute *FirstName* is m. The length of an ABF for a *FirstName* value, l^{FN}, can be calculated as [31]:

$$l^{FN} = \frac{1}{1 - \sqrt[km]{\rho}}. \tag{A.6}$$

If ρ is set to 0.5 then all the bits in the ABF are set to 1 with 0.5 probability, i.e. half of the bits in a given ABF will be set to 1. This leads to a more uniform distribution of 1-bits across all QID attributes which will provide more security against frequency attacks [61].

2. **Eligible bit identification:** In order to make the RBFs resilient to frequency attacks, in this step certain bits from each ABF are selected to be sampled, using a statistical method [61]. The actual sampling of these bits will take place in the fourth step. This bit identification is conducted based on a frequency analysis of QID attribute values and 1-bit positions in the corresponding ABFs. For instance, assume that a certain set of bit positions, **P**, in a ABF is unique only to *City* QID attribute i.e. those bit positions are set by only the q-grams from *City* QID attribute. In such a scenario an adversary would be able to identify these bit positions by aligning the frequency distributions of the bit positions with the frequency distributions of q-grams from *City* values. To address this issue Durham et al. [61] proposed to set a security constraint c, which determines the number of QID attribute values whoese q-grams are mapped to each bit position. That is, only the bit positions that can be linked back to at least c QID attribute values are selected from each ABF to be included in RBF. With higher c better security can be obtained, but on the other hand a higher value of c incurs high computational cost since more QID attribute values from a record need to be encoded.

3. **QID attribute weighting:** In the third step of the RBF generation process weights for each QID attribute are decided to sample bits. Based on the discriminatory power of each QID, a DO can assign weights to each QID attribute and different number of bits will be selected from each ABF based on these weights. Note that according to the original methodology [61], sampling with replacement is allowed. However, in a situation where no knowledge of the discriminatory power of QID attributes is available to the DOs, Durham et al. [61] proposed to use the Fellegi and Sunter probabilistic QID attribute weighting approach as used in traditional record linkage [73, 94, 224] to calculate weights.

4. **RBF generation:** The final step is to sample bits from ABFs and generate the RBF. First, eligible bits selected in the second step are sampled using weights assigned to each QID in the third step. Then all the sampled bits from different ABFs are concatenated into a single BF. Finally, to improve the privacy even more, the bit positions of the concatenated BF are permuted and the resulting BF is the final RBF. It is important to note that the same permutation seed must be applied for all BFs from different databases. Therefore DOs in a PPRL protocol must agreed upon a seed for the permutation prior to the RBF generation process. This random shuffling of bits will prevent an adversary from predicting which bits in a RBF correspond to which ABF, based on their sequence.

In Figure A.2 we illustrate an example generation of a RBF following the above discussed four steps. Note that this is only a high level illustration which shows the main steps of the RBF generation process.

A.4 Bloom Filter Hardening Techniques

As we described in Chapters 4, 7, and 8, BF encoding is susceptible to different cryptanalysis attacks [42, 119, 147]. These attack exploit the vulnerabilities of BF encoding using various techniques such as pattern-mining and graph matching. To overcome these weaknesses hardening techniques have been proposed for BFs over the years. In the following we describe five such hardening techniques that can be used to improve the privacy guarantees of BF encoding.

A.4.1 Salting

In salting a record specific salt value is concatenated with the q-grams before hashing. Salting was proposed by Niedermeyer et al. [147] to avoid QID value based frequency attacks on BFs. By concatenating a record specific salt value to each q-gram, the same q-grams from different records can be hashed differently. The values selected as salts are often unique to individuals and do not change over time. For instance year of birth or the birth place of an individual can be a salt value.

Assume two records r_1 and r_2 with the same first name 'peter' but with two different birth years, '1985' and '1992', respectively. Since the q-gram sets from both r_1 and r_2 are the same, with encoding without salting those q-gram sets will be hashed into same bit positions. However, with salting using birth years, the q-gram sets of r_1 and r_2 will be different.

$$r_1 \longrightarrow \{pe1985,\ et1985,\ te1985,\ er1985\}$$

$$r_2 \longrightarrow \{pe1992,\ et1992,\ te1992,\ er1992\}$$

The resulting q-gram sets will be hashed into two different sets of bit positions. Furthermore, the frequency information of salted q-grams will be different from plaintext q-grams. As we discussed in Chapter 5, exploiting frequency vulnerability is a key aspect for most cryptanalysis attacks on BFs. With salting however, extracting and aligning frequency information will be difficult since frequent q-grams will become much less frequent.

A.4.2 Balancing

Some cryptanalysis attacks on BFs analyse Hamming weight distribution of BFs to identify possible bit patterns and align them with frequencies of q-grams in a plaintext database [43, 119]. However, BFs in an encoded database have a uniform distribution of Hamming weight, the identification of frequent bit patterns will become potentially more difficult.

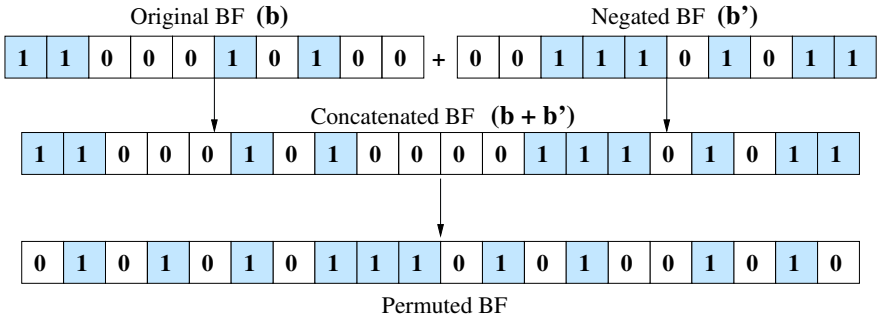


Figure A.3: An example of BF balancing. The original BF $\mathbf{b}$ is first negated to get the $\mathbf{b}'$. Then both the original and the negated copy of BF is concatenated, $(\mathbf{b} + \mathbf{b}')$, and randomly permuted.

Balanced codes, also referred to as constant-weight codes, are used as a binary forward error correction mechanism where binary vector contains equal number of 1-bits and 0-bits [113]. As shown in [21], the Hamming weight distribution of a set of binary vectors can be made uniform by joining each binary vector with its negated copy. This concept was employed by Schnell and Borgs [180] to propose balanced BFs for PPRL. In balanced BFs a negated copy of a BF of length l is concatenated with the original BF resulting in a new bit vector of length $2l$. In order to improve the privacy Schnell and Borgs also proposed to permute the concatenated bit vector [180].

When balancing is applied the resulting BF (bit vector) will have same number of 1-bits and 0-bits. In Figure A.3 we show an example balancing process where a BF $\mathbf{b}$ of length $l = 10$ is balanced using its negated copy, $\mathbf{b}'$. The length of final BF is $l = 20$. It is worth noting that the increased length of balanced BFs and their uniform hamming weight will increase the runtime and memory requirements of the PPRL process. The cryptanalysis attack proposed by Christen et al. [42] was however shown to be able to attack BFs successfully even when the BFs has been balanced.

A.4.3 XOR-Folding

In chemical informatics, a folding of a bit vector is generally used to improve the efficiency of computations such as searching [34]. Same folding techniques have also been used in PPRL applications to efficiently link records [118]. Motivated by this work, Schnell and Borgs [181] have proposed a hardening technique named XOR-folding for BF encoding to improve its privacy guarantees.

In XOR-folding a BF, $\mathbf{b}$, of length l is first divided into two halves, $\mathbf{b}_1$ and $\mathbf{b}_2$, of length $l/2$. $\mathbf{b}_1$ contains the first half of bit positions in $\mathbf{b}$ ranging from $[1, ..., l/2]$ whereas $\mathbf{b}_2$ contains the second half of bit positions ranging from $[l/2 + 1, ..., l]$. Then, the two BFs $\mathbf{b}_1$ and $\mathbf{b}_2$ are combined into a new BF $\mathbf{b}_{fold}$ of length $l/2$ using the bit-wise XOR operation, $\oplus$:

$$\mathbf{b}_{fold}[i] = \mathbf{b}_1[i] \oplus \mathbf{b}_2[i], \tag{A.7}$$

where $0 \leq i < l/2$. For instance, assume a BF, $\mathbf{b}$, of length $l = 10$ bits: $[0,1,1,0,0,1,0,1,1,0]$. After dividing $\mathbf{b}$ into two halfs we get $\mathbf{b}_1 = [0,1,1,0,0]$ and $\mathbf{b}_2 = [1,0,1,1,0]$. $\mathbf{b}_1$ and $\mathbf{b}_1$ are then combined using bit-wise XOR operation which results the folded BF, $\mathbf{b}_{fold} = [1,1,0,1,0]$. Due to the combination of two bit positions into one using XOR operation, it is difficult to recover any frequency information from the folded BF, thus improving privacy. According to Schnell and Borgs [181], folding can be done several times depending on the user preference. However, XOR folding will likely also lead to a reduction of linkage quality because information about two bit positions are combined into one resulting a shorter BF.

A.4.4 Window based XORing

Even though the hardening techniques we discussed above can improve privacy of the BF encoding, they can also potentially reduce the linkage quality. To address this issue Ranbaduge and Schnell [161] have recently proposed two hardening techniques for BFs. One of those technique is window based XORing where a sliding window based approach is used to harden the original BF.

In window based XORing, two sliding windows $\mathbf{w}_1$ and $\mathbf{w}_2$ with a window size w are progressed along a BF $\mathbf{b}$ of length l. Essentially, $\mathbf{w}_1$ and $\mathbf{w}_2$ are two lists of bit positions extracted from $\mathbf{b}$ where in each iteration $\mathbf{w}_1$ starts at the position p with $1 \leq p \leq l - w$, and $\mathbf{w}_2$ starts at the position $(p+1) mod\ l$ with *mod* being the modulo function. In each iteration the two windows $\mathbf{w}_1$ and $\mathbf{w}_2$ are XORed to get a single list of bit positions of size w. The XORed list is finally used to replace the bit values in $\mathbf{b}$ that corresponds to the window $\mathbf{w}_1$. After each iteration both sliding windows are moved by 1 bit position for the next iteration. This process is continued till all the bit values in $\mathbf{b}$ are processed.

According to Ranbaduge and Schnell [161], with a small window size w, such as $w = [2,3,4]$, the window based XORing requires longer runtime, but provides accurate similarity calculations. On the other hand, with a large window size w, such as $w = [400,500]$ (assuming BF length $l = 1,000$), the hardening process is more efficient but will result in less accurate similarities.

A.4.5 Re-sampling based XORing

The second hardening method proposed by Ranbaduge and Schnell [161] is re-sampling based XORing where a random sampling based technique is used.

For a BF $\mathbf{b}$ of length l, random sampling with replacement is done l times, where in each sampling step two random bit positions p_1 and p_2 are selected with $1 \leq p_1, p_2 \leq l$. Using the sampled bit positions, the bit values $\mathbf{b}[p_1]$ and $\mathbf{b}[p_2]$ are extracted from the BF $\mathbf{b}$ and XORed to get a single bit value. This process is continued l times and the resulting l XORed bit values are then concatenated together to get the hardened BF. Ranbaduge and Schnell [161] argued that if the original BF $\mathbf{b}$ has more 1 bits than 0 bits, the hardened BF will have more 0 bits due the XORing process and that will lead to lower similarity calculations.

Appendix B

Assessing Disclosure Risk in Microdata

In this appendix we describe the main types of disclosure risks, as well as disclosure risk measures that have been proposed in the contexts of both data publishing and statistical disclosure control.

B.1 Types of Disclosure Risks

A disclosure takes place when a person or a system learns sensitive information about an individual or a quasi-identifying (QID) attribute value which is not meant to be learned [71, 192]. There are two main types of disclosure risks: *attribute disclosure risk* and *identity disclosure risk* [9, 58]. Attribute disclosure risk measures the risk of reidentifying one or more QID attribute value(s) in a sensitive database. In attribute disclosure, the adversary is able to reidentify the plaintext value(s), such as first name or street address, that have been encoded. However, attribute disclosure does not necessarily imply that the adversary is capable of reidentifying the actual real-world entities represented by these QID attribute values.

Identity disclosure, on the other hand, is defined as the risk of reidentifying actual real-world entities. If an adversary is able to reidentify individual entities (usually people in the context of PPRL) in an encoded database using a privacy attack then their identities are compromised [192].

B.2 Assessing Disclosure Risk

Three disclosure risk measures for publishing of microdata have been proposed by Truta et al. [195] to analyse different disclosure control techniques. The three disclosure risk measures are minimal disclosure risk, maximal disclosure risk, and weighted disclosure risk. The minimal disclosure risk measures the ratio of unique records (records that have a unique set of QID attribute values) in an encoded database. The maximal disclosure risk measures the ratio of distinct record clusters (groups of records based on their sets of QID attribute values) in the encoded

database. The weighted disclosure risk uses a weighting approach to increase the importance of unique values in a record over the rest of the values in that record, and to increase the importance of records with smaller frequencies (calculated based on a set of QID attribute values) over rest of the records with high frequencies in the encoded database [195].

A set of empirical disclosure risk measures for microdata and tabular data was proposed by Domingo-Ferrer and Torra [55]. Unlike most other measures which are based on the uniqueness of records, these proposed measures are based on how records between a plaintext and an encoded database can be linked. In the first method, the records are linked based on distances calculated between records in the plaintext and encoded databases. In the second method, the authors used probabilistic record linkage to find matching record pairs across the plaintext and encoded databases [55]. The matching of records is solved as a linear sum assignment problem. For both of these methods, the percentages of correctly linked encoded and plaintext record pairs are considered as measures of disclosure risk. In the last method, named interval disclosure, the authors first ranked values in each QID attribute independently based on their frequencies, and then a rank interval is defined around each value. Using a threshold which is defined to measure the differences between ranks, one can then align plaintext values and encoded values in ranked QID attributes based on their intervals. The proportion of plaintext values in a given QID attribute which fall into the intervals of their corresponding encoded values is considered as a measure of disclosure risk for that QID attribute.

A disclosure risk assessment via record linkage was proposed by Domingo-Ferrer et al. [53] assuming a maximum-knowledge adversary. The authors have considered an adversary who has access to both the plaintext and corresponding encoded databases along with the details of all QID attribute values of the entities in those databases. The authors assumed that the adversary aims to link records across the encoded and plaintext databases to reidentify encoded records. To assess the disclosure risk in such a scenario, first the distribution of distances between record pairs across plaintext and encoded databases is calculated. This distribution is then compared with another distribution of distances that is known to correspond to a database with no disclosure risks. The more similar the two distributions, the more difficult it will be for the adversary to correctly link records between encoded and plaintext databases.

In a recent survey, Taylor et al. [192] discussed five measures to assess disclosure risks of microdata before being released for analytical purposes. These measures assume that the adversary has access to a global population where the microdata represent a subset (a sample) of entities in this population, such that each record in the sensitive microdata can be matched with a record in the population. The discussed five measures are: (1) the expected number of population uniques, (2) the expected number of sample uniques that are population uniques, (3) the expected number of correct matches among sample uniques, (4) the probability of a correct match given a unique match, and (5) the probability of a correct match. Here a match in the third, fourth, and fifth measures is referred to a record in the microdata

and a record in the population, both having the same set of values for a given set of QID attributes.

B.3 Disclosure Control Methods

Different data anonymisation methods can be used to control the disclosure risks of values in a sensitive database. These anonymisation methods either mask or alter the QID attribute values in records using different techniques in order to minimise the risk of reidentifying those QID attribute values or individuals.

k-anonymity is a popular data anonymisation technique used in both privacy-preserving data mining and PPRL applications to control the disclosure risks of values in a sensitive database [101, 104, 141, 191]. In *k*-anonymity, the anonymisation of records is conducted by generalising or masking certain QID attribute values. In generalisation, individual QID attribute values in the database are replaced by a broader category (for example date of birth is replaced by year of birth), while in masking certain sensitive and unique QID attribute values are replaced by a specific character, such as '*'. A database can be described as *k*-anonymised if an individual entity represented by a record cannot be distinguished from at least $k-1$ other individual entities whose records also appear in the same database. However, research has shown that *k*-anonymised databases can be susceptible to certain types of privacy attacks such as background knowledge attacks and homogeneity attacks [132, 136]. To address the limitations and privacy concerns of *k*-anonymity [136, 217], different improved generalisation methods have been proposed such as *p*-sensitive *k*-anonymity, *l*-diversity, and *t*-closeness [132, 136, 196]. These methods aim to introduce diversity among the QID attribute values grouped using *k*-anonymity by reducing the granularity of data.

Differential privacy is a mathematical concept proposed to control disclosure risk by allowing the implementation of techniques that provide strong privacy guarantees against record reidentification [66]. Differential privacy aims to address the problem of delivering useful information about a population while revealing no harmful information about the individuals in that population [66, 133]. Differential privacy is usually achieved by introducing certain amounts of random noise into the results of queries on sensitive data. By adding noise to the query results, an adversary who observes those results will not be able to uniquely identify sensitive information of an individual entity [133]. Methods such as Laplace noise [67] and randomised response [218] have been used to add noise to data in a probabilistic way. Recently differential privacy has been employed in PPRL applications to ensure the privacy of entities whose information are stored in a sensitive database [32, 97].

References

1. Abramowitz, M., and Stegun, I. A. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*, vol. 55. US Government printing office, 1964. (cited on page 26)

2. Acquisti, A., Friedman, A., and Telang, R. Is there a cost to privacy breaches? an event study. In *International Conference on Information Systems (ICIS)* (January 2006), p. 94. (cited on page 2)

3. Aggarwal, C., and Wang, H. *Managing and Mining Graph Data*, vol. 40 of *Advances in Database Systems*. Springer, 2010. (cited on pages 90, 165, 170, and 171)

4. Agrawal, R., and Srikant, R. Fast algorithms for mining association rules in large databases. In *International Conference on Very Large Data Bases (VLDB)* (San Francisco, CA, USA, 1994), Morgan Kaufmann Publishers Inc., pp. 487–499. (cited on page 135)

5. Al-Lawati, A., Lee, D., and McDaniel, P. Blocking-aware private record linkage. In *International Workshop on Information Quality in Information Systems (IQIS)* (Baltimore, 2005), pp. 59–68. (cited on pages 14 and 30)

6. Al-Lawati, A., Lee, D., and McDaniel, P. Blocking-aware private record linkage. In *International Workshop on Information Quality in Information Systems* (New York, NY, USA, 2005), ACM, pp. 59–68. (cited on page 19)

7. Alaggan, M., Gambs, S., and Kermarrec, A.-M. BLIP: Non-interactive differentially-private similarity computation on Bloom filters. In *Symposium on Self-Stabilizing Systems* (Toronto, 2012), pp. 202–216. (cited on page 27)

8. Andoni, A., and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM 51* (2008). (cited on page 171)

9. Andreou, A., Goga, O., and Loiseau, P. Identity vs. attribute disclosure risks for users with multiple social profiles. In *IEEE/ACM International Conference on Advances in Social Network Analysis and Mining (ASONAM)* (Sydney, Australia, 2017), pp. 163–170. (cited on pages 55 and 225)

10. Antoni, M., and Schnell, R. The past, present and future of the german record linkage center. *Journal of Economics and Statistics* (2017). (cited on pages 4, 8, 23, 129, 160, 183, and 213)

11. Arora, S., and Barak, B. *Computational Complexity: A Modern Approach*, 1 ed. Cambridge University Press, 2009. (cited on page 33)

12. Aumann, Y., and Lindell, Y. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography Conference (TCC)* (Amsterdam, 2007), Springer, pp. 137–156. (cited on page 21)

13. Ayyagari, R. An exploratory analysis of data breaches from 2005-2011: Trends and insights. *Journal of Information Privacy and Security 8*, 2 (2012), 33–56. (cited on page 2)

14. Bachteler, T., Schnell, R., and Reiher, J. An empirical comparison of approaches to approximate string matching in private record linkage. In *Statistics Canada Symposium* (January 2010), pp. 290–295. (cited on pages 1 and 17)

15. Baer, M. Corporate policing and corporate governance: What can we learn from Hewlett-Packard's pretexting scandal? *New York University Public Law and Legal Theory Working Papers 77* (December 2008). (cited on page 50)

16. Bansal, G. Distinguishing between privacy and security concerns: An empirical examination and scale validation. *Journal of Computer Information Systems 57*, 4 (2017), 330–343. (cited on page 49)

17. Barla, A., Odone, F., and Verri, A. Histogram intersection kernel for image classification. In *IEEE International Conference on Image Processing (ICIP)* (2003), vol. 3, pp. III–513. (cited on page 111)

18. Bayardo, R. J. Efficiently mining long patterns from databases. *ACM SIGMOD Record 27*, 2 (1998), 85–93. (cited on page 136)

19. Bellare, M., Canetti, R., and Krawczyk, H. Keying hash functions for message authentication. In *Annual International Cryptology Conference* (1996), Springer, pp. 1–15. (cited on pages 107 and 161)

20. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., and Widom, J. Swoosh: A generic approach to entity resolution. *The International Journal on Very Large Data Bases 18*, 1 (2009), 255–276. (cited on page 1)

21. Berger, J. M. A note on error detection codes for asymmetric channels. *Information and Control 4*, 1 (1961), 68–73. (cited on page 223)

22. Bilenko, M., and Mooney, R. J. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Washington DC, 2003), pp. 39–48. (cited on page 17)

23. Bizer, C., Heath, T., and Berners-Lee, T. Linked data - the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS) 5*, 3 (2009), 1–22. (cited on page 19)

24. Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13*, 7 (1970), 422–426. (cited on pages 23, 129, 133, and 213)

25. Bondy, A., and Murty, U. *Graph Theory*. Springer, London, 2011. (cited on page 171)

26. Bonomi, L., Xiong, L., Chen, R., and Fung, B. Frequent grams based embedding for privacy preserving record linkage. In *ACM International Conference on Information and Knowledge Management (CIKM)* (Maui, Hawaii, 2012), pp. 1597–1601. (cited on pages 2, 8, 15, and 27)

27. Borgs, C. *Optimal Parameter Choice for Bloom Filter-based Privacy-Preserving Record Linkage*. PhD thesis, University of Duisburg-Essen, Germany, 2019. (cited on pages 17, 153, 178, 190, and 215)

28. Bose, P., Guo, H., Kranakis, E., Maheshwari, A., Morin, P., Morrison, J., Smid, M., and Tang, Y. On the false-positive rate of Bloom filters. *Information Processing Letters 108* (2008), 210–213. (cited on page 214)

29. Boyd, J. H., Randall, S. M., and Ferrante, A. M. *Application of Privacy-Preserving Techniques in Operational Record Linkage Centres*. Springer, Cham, 2015, pp. 267–287. (cited on pages 4, 8, 23, 81, 129, 160, 183, and 213)

30. Brandes, U. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology 25*, 2 (2001), 163–177. (cited on page 177)

31. Broder, A., and Mitzenmacher, M. Network applications of Bloom filters: A survey. In *Internet Mathematics* (2004). (cited on pages 214 and 220)

32. Cao, J., Rao, F.-Y., Bertino, E., and Kantarcioglu, M. A hybrid private record linkage scheme: Separating differentially private synopses from matching records. In *IEEE International Conference on Data Engineering* (April 2015), pp. 1011–1022. (cited on pages 28 and 227)

33. Chen, F., Jiang, X., Wang, S., Schilling, L. M., Meeker, D., Ong, T., Matheny, M. E., Doctor, J. N., Ohno-Machado, L., and Vaidya, J. Perfectly secure and efficient two-party electronic-health-record linkage. *IEEE Internet Computing 22*, 2 (March 2018), 32–41. (cited on pages 19 and 28)

34. Chen, J., Swamidass, S. J., Dou, Y., Bruand, J., and Baldi, P. Chemdb: A public database of small molecules and related chemoinformatics resources. *Bioinformatics 21*, 22 (2005), 4133–4139. (cited on page 223)

35. Christen, P. Privacy-preserving data linkage and geocoding: Current approaches and research directions. In *IEEE International Conference on Data Mining - Workshops (ICDMW)* (Hong Kong, 2006), pp. 497–501. (cited on pages 18 and 19)

36. Christen, P. Automatic training example selection for scalable unsupervised record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Osaka, 2008), vol. 5012, pp. 511–518. (cited on page 17)

37. Christen, P. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 24* (September 2011), 1537–1555. (cited on pages 1, 14, and 32)

38. Christen, P. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012. (cited on pages 1, 2, 13, 14, 16, 17, 23, 24, 26, 30, 31, 32, 84, 90, 164, 177, and 213)

39. Christen, P., and Goiser, K. Quality and complexity measures for data linkage and deduplication. In *Quality Measures in Data Mining* (2007), vol. 43 of *Studies in Computational Intelligence*, Springer, pp. 127–151. (cited on page 33)

40. Christen, P., Ranbaduge, T., and Schnell, R. *Linking Sensitive Data – Methods and Techniques for Practical Privacy-Preserving Information Sharing*. Springer, 2020. (cited on pages 1, 2, 3, 4, 8, 13, 14, 15, 17, 18, 19, 21, 22, 24, 27, 32, 37, 41, 43, 44, 47, 48, 52, 61, 116, 124, 202, 218, and 220)

41. Christen, P., Ranbaduge, T., Vatsalan, D., and Schnell, R. Precise and fast cryptanalysis for Bloom filter based privacy-preserving record linkage. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (October 2018). (cited on pages 23, 45, 52, 55, 74, 76, 82, 96, 97, 102, 108, 114, 129, 160, 163, 167, 187, 191, and 219)

42. Christen, P., Schnell, R., Vatsalan, D., and Ranbaduge, T. Efficient cryptanalysis of Bloom filters for privacy-preserving record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Jeju, Korea, 2017), Springer, pp. 628–640. (cited on pages 2, 4, 5, 6, 8, 20, 21, 23, 37, 48, 50, 54, 56, 61, 72, 73, 74, 75, 76, 96, 97, 102, 129, 187, 191, 200, 217, 219, 222, and 223)

43. Christen, P., Vidanage, A., Ranbaduge, T., and Schnell, R. Pattern-mining based cryptanalysis of Bloom filters for privacy-preserving record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Melbourne, 2018), Springer, pp. 628–640. (cited on pages 7, 56, 61, 66, 96, 97, 108, 129, 130, 136, 156, 157, 158, 186, 207, and 222)

44. Churches, T., and Christen, P. Some methods for blindfolded record linkage. *BMC Medical Informatics and Decision Making 4*, 9 (2004). (cited on pages 22, 24, and 53)

45. Churches, T., Christen, P., Lim, K., and Zhu, J. X. Preparation of name and address data for record linkage using hidden Markov models. *BMC Medical Informatics and Decision Making 2*, 9 (2002). (cited on pages 15 and 16)

46. Culnane, C., Rubinstein, B., and Teague, V. Vulnerabilities in the use of similarity tables in combination with pseudonymisation to preserve data privacy in the UK Office for National Statistics' privacy-preserving record linkage. *arXiv Preprint* (2017). (cited on pages 37, 52, 78, 82, 89, 96, and 97)

47. Culnane, C., Rubinstein, B. I., and Teague, V. Health data in an open world. *arXiv Preprint* (2017). (cited on pages 3, 38, 45, and 80)

48. de Winter, J., Gosling, S., and Potter, J. Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological Methods 21* (September 2016), 273–290. (cited on page 110)

49. Dean, J., and Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2008), 107–113. (cited on page 210)

50. Desai, T., Ritchie, F., and Welpton, R. Five safes: Designing data access for research. Tech. rep., Department of Accounting, Economics and Finance, Bristol Business School, University of the West of England, 2016. (cited on page 202)

51. Dillinger, P. C., and Manolios, P. Bloom filters in probabilistic verification. In *International Conference on Formal Methods in Computer-Aided Design* (2004), Springer, pp. 367–381. (cited on pages 23, 214, and 216)

52. Dillinger, P. C., and Manolios, P. Fast and accurate bitstate verification for spin. In *International SPIN Workshop on Model Checking of Software* (2004), Springer, pp. 57–75. (cited on pages 23, 187, 191, 199, and 216)

53. Domingo-Ferrer, J., Ricci, S., and Soria-Comas, J. Disclosure risk assessment via record linkage by a maximum-knowledge attacker. In *Annual Conference on Privacy, Security and Trust (PST)* (Los Alamitos, USA, 2015), pp. 28–35. (cited on pages 37, 40, 43, 53, and 226)

54. Domingo-Ferrer, J., and Torra, V. Disclosure risk assessment in statistical microdata protection via advanced record linkage. *Statistics and Computing 13*, 4 (2003), 343–354. (cited on page 19)

55. Domingo-Ferrer, J., and Torra, V. Disclosure risk assessment in statistical data protection. *Journal of Computational and Applied Mathematics 164–165* (2004), 285–293. (cited on page 226)

56. Dong, C., Chen, L., and Wen, Z. When private set intersection meets big data: An efficient and scalable protocol. In *ACM SIGSAC Conference on Computer & Communications Security* (Berlin, 2013), pp. 789–800. (cited on page 31)

57. Dougherty, C. R. Vulnerability note vu836068: MD5 vulnerable to collision attacks. https://www.kb.cert.org/vuls/id/836068, December 2008. (cited on page 24)

58. Duncan, G., and Lambert, D. The risk of disclosure for microdata. *Journal of Business & Economic Statistics 7*, 2 (1989), 207–217. (cited on pages 55 and 225)

59. Duncan, G. T., Elliot, M., and Gonzalez, J. J. S. *Statistical Confidentiality: Principles and Practice*. Springer, 2011. (cited on pages 6 and 82)

60. Dunn, H. L. Record linkage. *American Journal of Public Health 36*, 12 (1946), 1412–1416. (cited on page 13)

61. Durham, E., Kantarcioglu, M., Xue, Y., Toth, C., Kuzu, M., and Malin, B. Composite Bloom filters for secure record linkage. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 26*, 12 (2014), 2956–2968. (cited on pages 23, 52, 69, 74, 134, 144, 153, 178, 200, 219, 220, and 221)

62. Durham, E., Xue, Y., Kantarcioglu, M., and Malin, B. Private medical record linkage with approximate matching. *AMIA Annual Symposium Proceedings* (November 2010), 182–186. (cited on page 17)

63. Durham, E., Xue, Y., Kantarcioglu, M., and Malin, B. Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. *Information Fusion 13*, 4 (October 2012), 245—-259. (cited on pages 33, 34, and 82)

64. Durham, E. A. *A Framework for Accurate, Efficient Private Record Linkage*. PhD thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, TN, 2012. (cited on pages 15, 17, 18, 33, 82, 153, 178, 190, 215, and 219)

65. Dusserre, L., Quantin, C., and Bouzelat, H. A one way public key cryptosystem for the linkage of nominal files in epidemiological studies. *Medinfo 8* (1995), 644–647. (cited on pages 28 and 53)

66. Dwork, C. Differential privacy. *Automata, Languages and Programming* (2006), 1–12. (cited on pages 28, 30, 54, and 227)

67. Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography* (Berlin, Heidelberg, 2006), Springer, pp. 265–284. (cited on pages 83 and 227)

68. Dwork, C., and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science 9*, 3–4 (August 2014), 211––407. (cited on page 28)

69. Dwork, C., and Yekhanin, S. New efficient attacks on statistical disclosure control mechanisms. In *Annual International Cryptology Conference* (2008), Springer, pp. 469–480. (cited on page 37)

70. Edmonds, J., and Karp, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM) 19*, 2 (1972), 248–264. (cited on page 177)

71. Elliot, M., Mackey, E., and O'Hara, K. *The Anonymisation Decision-making Framework 2nd Edition: European practitioners' guide*. UK Anonymisation Network, Manchester, 2020. (cited on pages 40, 82, and 225)

72. Emmert-Streib, F., Dehmer, M., and Shi, Y. Fifty years of graph matching, network alignment and network comparison. *Information Sciences 346* (2016), 180–197. (cited on pages 165 and 171)

73. Fellegi, I. P., and Sunter, A. B. A theory for record linkage. *Journal of the American Statistical Association (JASA) 64*, 328 (1969), 1183–1210. (cited on pages 1, 13, 17, 25, 106, and 221)

74. Franke, M., Sehili, Z., Gladbach, M., and Rahm, E. Post-processing methods for high quality privacy-preserving record linkage. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology* (Cham, 2018), Springer, pp. 263–278. (cited on page 173)

75. Franke, M., Sehili, Z., Rohde, F., and Rahm, E. Evaluation of hardening techniques for privacy-preserving record linkage. In *International Conference on Extending Database Technology (EDBT)* (March 2021), ACM. (cited on page 202)

76. Fu, Z., Christen, P., and Boot, M. A supervised learning and group linking method for historical census household linkage. In *Australasian Data Mining Conference (AusDM), CRPIT* (Ballarat, Australia, 2011), vol. 121, pp. 153—-162. (cited on page 17)

77. Fung, B. C., Wang, K., Chen, R., and Yu, P. S. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys 42*, 4 (2010), 1–53. (cited on page 82)

78. Gale, D., and Shapley, L. College admissions and the stability of marriage. *The American Mathematical Monthly 69*, 1 (1962), 9–15. (cited on page 173)

79. Gentry, C. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC)* (New York, NY, USA, 2009), pp. 169—-178. (cited on pages 30 and 31)

80. Goldreich, O. Secure multi-party computation. Tech. rep., Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel, 2002. (cited on page 30)

81. Grahne, G., and Zhu, J. Fast algorithms for frequent itemset mining using FP-trees. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 17*, 10 (2005), 1347–1362. (cited on page 136)

82. Groeneveld, R. A., and Meeden, G. Measuring skewness and kurtosis. *Journal of the Royal Statistical Society: Series D (The Statistician) 33*, 4 (1984), 391–399. (cited on page 110)

83. Grover, A., and Leskovec, J. Node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), pp. 855—864. (cited on page 171)

84. Gu, L., and Baxter, R. *Decision Models for Record Linkage*. Springer, Berlin, Heidelberg, 2006, pp. 146—-160. (cited on page 14)

85. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., and Sharma, R. Discovering all most specific sentences. *ACM Transactions on Database Systems 28*, 2 (2003), 140–174. (cited on page 152)

86. Hadnagy, C. *Social engineering: The Art of Human Hacking*. John Wiley & Sons, 2010. (cited on page 49)

87. Hall, R., and Fienberg, S. Privacy-preserving record linkage. In *Privacy in Statistical Databases* (Corfu, Greece, 2010), Springer, pp. 269–283. (cited on pages 14, 15, 21, and 81)

88. Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*, 3 ed. Morgan Kaufmann, 2011. (cited on page 14)

89. Hand, D. J., and Christen, P. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing 28*, 3 (2018), 539–547. (cited on page 33)

90. He, X., Machanavajjhala, A., Flynn, C., and Srivastava, D. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)* (Dallas, 2017), pp. 1389–1406. (cited on pages 28 and 54)

91. Heimann, M., Lee, W., Pan, S., Chen, K.-Y., and Koutra, D. Hashalign: Hash-based alignment of multiple graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Melbourne, 2018), Springer, pp. 726–739. (cited on pages 165, 171, and 172)

92. Heimann, M., Shen, H., Safavi, T., and Koutra, D. Regal: Representation learning-based graph alignment. In *ACM International Conference on Information and Knowledge Management (CIKM)* (Torino, 2018), pp. 117—126. (cited on page 170)

93. Hershey, J. R., and Olsen, P. A. Approximating the kullback leibler divergence between gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2007), vol. 4, pp. IV–317–IV–320. (cited on page 111)

94. Herzog, T. N., Scheuren, F. J., and Winkler, W. E. *Data quality and record linkage techniques*. Springer, 2007. (cited on pages 1 and 221)

95. Hjaltason, G. R., and Samet, H. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 25*, 5 (2003), 530–549. (cited on page 27)

96. Inan, A., Kantarcioglu, M., Bertino, E., and Scannapieco, M. A hybrid approach to private record linkage. In *IEEE International Conference on Data Engineering (ICDE)* (Mexico, 2008), pp. 496–505. (cited on pages 8 and 30)

97. Inan, A., Kantarcioglu, M., Ghinita, G., and Bertino, E. Private record matching using differential privacy. In *International Conference on Extending Database Technology (EDBT)* (New York, NY, USA, 2010), ACM, pp. 123–134. (cited on pages 20 and 227)

98. Iwama, K., and Miyazaki, S. A survey of the stable marriage problem and its variants. In *International Conference on Informatics Education and Research for Knowledge-circulating Society (ICKS)* (2008), IEEE, pp. 131–136. (cited on page 177)

99. Jaffri, A., Glaser, H., and Millard, I. URI identity management for semantic web data integration and linkage. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops* (Berlin, Heidelberg, 2007), Springer, pp. 1125–1134. (cited on page 1)

100. Jakobsen, T. A fast method for the cryptanalysis of substitution ciphers. *Cryptologia 19*, 3 (1995), 265–274. (cited on page 72)

101. Kantarcioglu, M., Inan, A., Jiang, W., and Malin, B. Formal anonymity models for efficient privacy-preserving joins. *Data & Knowledge Engineering 68*, 11 (2009), 1206–1223. (cited on page 227)

102. Karakasidis, A., and Verykios, V. Secure blocking + secure matching = secure record linkage. *Journal of Computing Science and Engineering 5*, 3 (2011), 223–235. (cited on page 17)

103. Karakasidis, A., and Verykios, V. S. Privacy preserving record linkage using phonetic codes. In *Fourth Balkan Conference in Informatics* (Thessaloniki, Greece, 2009), IEEE, pp. 101–106. (cited on pages 17 and 33)

104. Karakasidis, A., and Verykios, V. S. Reference table based k-anonymous private blocking. In *ACM Symposium on Applied Computing* (New York, NY, USA, 2012), pp. 859–864. (cited on pages 14, 26, 82, and 227)

105. Karakasidis, A., Verykios, V. S., and Christen, P. Fake injection strategies for private phonetic matching. In *Data Privacy Management and Autonomous Spontaneus Security* (Berlin, Heidelberg, 2012), Springer, pp. 9–24. (cited on pages 33 and 34)

106. Karapiperis, D., Gkoulalas-Divanis, A., and Verykios, V. S. Distance-aware encoding of numerical values for privacy-preserving record linkage. In *IEEE International Conference on Data Engineering (ICDE)* (San Diego, 2017), pp. 135–138. (cited on pages 8, 29, 54, and 210)

107. Karapiperis, D., Gkoulalas-Divanis, A., and Verykios, V. S. A framework for distance-aware privacy-preserving record linkage. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 30*, 2 (2017), 292–304. (cited on pages 8 and 29)

108. Karapiperis, D., Gkoulalas-Divanis, A., and Verykios, V. S. Fast schemes for online record linkage. *Data Mining and Knowledge Discovery (DMKD) 32*, 5 (2018), 1229–1250. (cited on page 129)

109. Karapiperis, D., and Verykios, V. S. An LSH-based blocking approach with a homomorphic matching technique for privacy-preserving record linkage. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2014). (cited on pages 8, 19, and 30)

110. Karmel, R. Data linkage protocols using a statistical linkage key. *Australian Institute of Health and Welfare*, CS1 (2005). (cited on pages 26, 95, 98, and 102)

111. Katz, J., and Lindell, Y. *Introduction to modern cryptography*. CRC press, 2007. (cited on pages 24, 53, and 84)

112. Kirsch, A., and Mitzenmacher, M. Less hashing, same performance: Building a better Bloom filter. In *European Symposium on Algorithms* (2006), pp. 456–467. (cited on page 216)

113. Knuth, D. Efficient balanced codes. *IEEE Transactions on Information Theory 32*, 1 (1986), 51–53. (cited on page 223)

114. Knuth, D. E. Big omicron and big omega and big theta. *ACM SIGACT News 8*, 2 (1976), 18–24. (cited on page 54)

115. Krawczyk, H., Bellare, M., and Canetti, R. HMAC: Keyed-hashing for message authentication. In *Internet RFCs* (1997). (cited on pages 25, 77, and 78)

116. Kriege, N. M., Johansson, F. D., and Morris, C. A survey on graph kernels. *Applied Network Science 5*, 6 (2019). (cited on page 171)

117. Krishnamurthy, B., and Wills, C. E. On the leakage of personally identifiable information via online social networks. In *ACM workshop on Online social networks* (2009), pp. 7–12. (cited on page 2)

118. Kristensen, T. G., Nielsen, J., and Pedersen, C. N. S. A tree based method for the rapid screening of chemical fingerprints. In *Algorithms in Bioinformatics* (Berlin, Heidelberg, 2009), Springer, pp. 194–205. (cited on page 223)

119. Kroll, M., and Steinmetzer, S. Automated cryptanalysis of Bloom filter encryptions of databases with several personal identifiers. In *International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC)* (Lisbon, 2015), pp. 341–356. (cited on pages 2, 4, 5, 6, 8, 20, 21, 23, 50, 52, 54, 71, 72, 96, 97, 108, 129, 163, 167, 186, 205, 216, 218, and 222)

120. Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 1-2 (1955), 83–97. (cited on page 174)

121. Kuzu, M., Kantarcioglu, M., Durham, E., and Malin, B. A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. In *Privacy Enhancing Technologies* (Waterloo, 2011), pp. 226–245. (cited on pages 2, 4, 5, 6, 8, 20, 37, 40, 48, 52, 54, 66, 67, 68, 69, 82, 96, 97, 129, 163, 167, 200, 205, and 218)

122. Kuzu, M., Kantarcioglu, M., Durham, E., Toth, C., and Malin, B. A practical approach to achieve private medical record linkage in light of public resources. *Journal of the American Medical Informatics Association (JAMIA) 20*, 2 (2013), 285–292. (cited on pages 68 and 69)

123. Kuzu, M., Kantarcioglu, M., Inan, A., Bertino, E., Durham, E., and Malin, B. Efficient privacy-aware record integration. In *International Conference on Extending Database Technology (EDBT)* (New York, NY, USA, 2013), ACM, pp. 167—-178. (cited on pages 8 and 30)

124. Lai, P. K., Yiu, S.-M., Chow, K., Chong, C., and Hui, L. C. K. An efficient Bloom filter based solution for multiparty private matching. In *International Conference on Security and Management* (2006), pp. 286–292. (cited on pages 20 and 214)

125. Lê Cao, K.-A., Martin, P. G., Robert-Granié, C., and Besse, P. Sparse canonical methods for biological data integration: Application to a cross-platform study. *BMC Bioinformatics 10*, 34 (January 2009). (cited on page 1)

126. Lenzerini, M. Data integration: A theoretical perspective. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (New York, NY, USA, 2002), pp. 233–246. (cited on page 1)

127. Leskovec, J., Rajaraman, A., and Ullman, J. D. *Mining of Massive Datasets*. Cambridge University Press, 2014. (cited on pages 25 and 31)

128. Leurent, G., and Peyrin, T. SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust. In *USENIX Security Symposium* (2020), USENIX Association, pp. 1839–1856. (cited on page 24)

129. Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady 10*, 8 (1966), 707–710. (cited on pages 27 and 91)

130. Levina, E., and Bickel, P. The Earth Mover's distance is the Mallows distance: Some insights from statistics. In *IEEE International Conference on Computer Vision (ICCV)* (2001), vol. 2, pp. 251–256. (cited on page 110)

131. Li, F., Chen, Y., Luo, B., Lee, D., and Liu, P. Privacy preserving group linkage. In *Scientific and Statistical Database Management* (Portland, 2011), pp. 432–450. (cited on page 21)

132. Li, N., Li, T., and Venkatasubramanian, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE International Conference on Data Engineering (ICDE)* (2007), pp. 106–115. (cited on page 227)

133. Li, N., Lyu, M., Su, D., and Yang, W. *Differential Privacy: From Theory to Practice*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan and Claypool Publishers, 2017. (cited on page 227)

134. Liginlal, D., Sim, I., and Khansa, L. How significant is human error as a cause of privacy breaches? an empirical study and a framework for error management. *Computers and Security 28* (May 2009), 215–228. (cited on page 50)

135. Lindell, Y., and Pinkas, B. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality (JPC) 1*, 1 (2009), 59—-98. (cited on pages 20, 21, 22, and 30)

136. Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (March 2007). (cited on page 227)

137. Mahajan, P., and Sachdeva, A. A study of encryption algorithms AES, DES and RSA for security. *Global Journal of Computer Science and Technology (GJCST)* (2013). (cited on page 24)

138. Martin, K. M. *Everyday Cryptography: Fundamental Principles and Applications*. Oxford University Press, 2012. (cited on page 161)

139. Mitchell, W., Dewri, R., Thurimella, R., and Roschke, M. A graph traversal attack on Bloom filter-based medical data aggregation. *International Journal of Big Data Intelligence (IJBDI) 4*, 4 (2017), 217–226. (cited on pages 37, 43, 45, 50, 76, 77, 82, 96, 97, 129, 167, 187, 191, and 205)

140. Mitzenmacher, M., and Upfal, E. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005. (cited on pages 133, 213, and 215)

141. Mohammed, N., Fung, B. C. M., and Debbabi, M. Anonymity meets game theory: Secure data integration with malicious participants. *The International Journal on Very Large Data Bases 20*, 4 (August 2011), 567–588. (cited on pages 21 and 227)

142. Montgomery, D., Peck, E., and Vining, G. *Introduction to Linear Regression Analysis*, 5 ed. Wiley, 2012. (cited on page 178)

143. Mooney, C. Z. *Monte Carlo Simulation*. SAGE Publications Inc., Thousand Oaks, United States, 1997. (cited on page 67)

144. Narayanan, A., and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy* (2008), pp. 111–125. (cited on pages 38, 45, and 80)

145. Naumann, F., and Herschel, M. *An introduction to duplicate detection*. Synthesis Lectures on Data Management. Morgan and Claypool Publishers, 2010. (cited on page 17)

146. Newcombe, H., Kennedy, J., Axford, S., and James, A. Automatic linkage of vital records. *Science 130*, 3381 (1959), 954–959. (cited on page 13)

147. Niedermeyer, F., Steinmetzer, S., Kroll, M., and Schnell, R. Cryptanalysis of basic Bloom filters used for privacy preserving record linkage. *Journal of Privacy and Confidentiality (JPC) 6*, 2 (2014), 59–79. (cited on pages 4, 5, 6, 23, 37, 48, 50, 52, 61, 69, 70, 71, 96, 97, 129, 160, 167, 200, 205, 216, 218, and 222)

148. Nissenbaum, H. Protecting privacy in an information age: The problem of privacy in public. *Law and Philosophy 17*, 5/6 (1998), 559–596. (cited on page 15)

149. O'Keefe, C., Yung, M., Gu, L., and Baxter, R. Privacy-preserving data linkage protocols. In *ACM Workshop on Privacy in the Electronic Society* (Washington DC, 2004), pp. 94–102. (cited on page 20)

150. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT* (Berlin, Heidelberg, 1999), Springer, pp. 223–238. (cited on pages 30 and 31)

151. Pang, C., Gu, L., Hansen, D., and Maeder, A. Privacy-preserving fuzzy matching using a public reference table. *Intelligent Patient Management* (2009), 71–89. (cited on pages 26 and 54)

152. Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., and Yang, D. H-mine: Hyper-structure mining of frequent patterns in large databases. In *IEEE International Conference on Data Mining (ICDM)* (San Jose, 2001), pp. 441–448. (cited on page 136)

153. Pita, R., Pinto, C., Sena, S., Fiaccone, R., Amorim, L., Reis, S., Barreto, M. L., Denaxas, S., and Barreto, M. E. On the accuracy and scalability of probabilistic data linkage over the Brazilian 114 million cohort. *IEEE Journal of Biomedical and Health Informatics (JBHI) 22*, 2 (2018), 346–353. (cited on pages 8, 23, 129, 183, and 213)

154. Pow, C., Iron, K., et al. Privacy-preserving record linkage: An international collaboration between Canada, Australia and Wales. *International Journal of Population Data Science (IJPDS) 1*, 1 (2017). (cited on pages 129, 160, and 213)

155. Quantin, C., Bouzelat, H., Allaert, F. A., Benhamiche, A., Faivre, J., and Dusserre, L. How to ensure data quality of an epidemiological follow-up: Quality assessment of an anonymous record linkage procedure. *International Journal of Medical Informatics 49*, 1 (1998), 117–122. (cited on pages 2 and 24)

156. Quantin, C., Bouzelat, H., Allaert, F.-A., Benhamiche, A.-M., Faivre, J., and Dusserre, L. Automatic record hash coding and linkage for epidemiological follow-up data confidentiality. *Methods of Information in Medicine 37*, 3 (1998), 271–277. (cited on page 24)

157. Ranbaduge, T. *A Scalable Blocking Framework for Multidatabase Privacy-Preserving Record Linkage*. PhD thesis, The Australian National University, Canberra, Australia, 2017. (cited on pages 14, 17, 18, 20, and 30)

158. Ranbaduge, T., Christen, P., and Schnell, R. Secure and accurate two-step hash encoding for privacy-preserving record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Singapore, 2020), pp. 139–151. (cited on pages 17, 25, 26, 37, 44, 85, 95, 98, 100, 101, 163, 164, 167, 176, 178, 184, 186, 190, and 208)

159. Ranbaduge, T., Christen, P., and Vatsalan, D. Tree based scalable indexing for multi-party privacy-preserving record linkage. In *Australasian Data Mining Conference (AusDM), CRPIT* (Brisbane, Australia, 2014), vol. 158. (cited on pages 4, 14, 17, 20, and 30)

160. Ranbaduge, T., Christen, P., and Vatsalan, D. Clustering-based scalable indexing for multi-party privacy-preserving record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (Hanoi, 2015), Springer. (cited on page 17)

161. Ranbaduge, T., and Schnell, R. Securing Bloom filters for privacy-preserving record linkage. In *ACM International Conference on Information and Knowledge Management (CIKM)* (October 2020), pp. 2185—2188. (cited on pages 160, 190, 197, 200, and 224)

162. Ranbaduge, T., Vatsalan, D., and Christen, P. Secure multi-party summation protocols: Are they secure enough under collusion? *Transactions on Data Privacy (TDP) 13* (2020), 25–60. (cited on pages 21, 44, and 51)

163. Randall, S., Brown, A., Ferrante, A., and Boyd, J. Privacy preserving linkage using multiple match-keys. *International Journal of Population Data Science (IJPDS) 4*, 1 (May 2019). (cited on pages 7, 25, 37, 44, 53, 85, 94, 95, 98, 102, 105, 106, 107, 108, 109, 114, 116, 126, 185, 190, and 207)

164. Randall, S. M., Ferrante, A. M., Boyd, J. H., Bauer, J. K., and Semmens, J. B. Privacy-preserving record linkage on large real world datasets. *Journal of Biomedical Informatics 50* (2014), 205–212. (cited on pages 19, 129, 160, 183, and 213)

165. Randall, S. M., Ferrante, A. M., Boyd, J. H., and Semmens, J. B. The effect of data cleaning on record linkage quality. *BMC Medical Informatics and Decision Making* (2013). (cited on pages 14 and 16)

166. Rao, F.-Y., Cao, J., Bertino, E., and Kantarcioglu, M. Hybrid private record linkage: Separating differentially private synopses from matching records. *ACM Transactions on Privacy and Security (TOPS) 22*, 3 (2019), 1–36. (cited on pages 28, 29, and 54)

167. Roesch, A. Matching data using sounds-like operators and sas® compare functions. In *SAS Global Forum* (2012), vol. 2012, Citeseer, pp. 1–11. (cited on pages 78 and 79)

168. Ruggles, S., Fitch, C. A., and Roberts, E. Historical census record linkage. *Annual Review of Sociology 44* (2018), 19–37. (cited on page 13)

169. Russell, R., and Odell, M. The soundex coding system. US Patent 1261167, April 1918. (cited on page 13)

170. Russell, S. J., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 2 ed. Prentice Hall, 2002. (cited on page 68)

171. Samarati, P. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE) 13*, 6 (November 2001), 1010—1027. (cited on pages 30 and 54)

172. Samarati, P., and Sweeney, L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Tech. rep., Computer Science Laboratory, SRI International, 1998. (cited on page 54)

173. Scannapieco, M., Figotin, I., Bertino, E., and Elmagarmid, A. K. Privacy preserving schema and data matching. In *ACM SIGMOD International Conference on Management of Data* (Beijing, China, 2007), pp. 653–664. (cited on pages 8, 27, 30, and 54)

174. Schmidlin, K., Clough-Gorr, K. M., and Spoerri, A. Privacy preserving probabilistic record linkage (p3rl): A novel method for linking existing health-related data and maintaining participant confidentiality. *BMC Medical Research Methodology 15* (May 2015), 46. (cited on page 19)

175. Schneier, B. *Applied cryptography: Protocols, algorithms, and source code in C*, 2 ed. John Wiley & Sons, New York, NY, USA, 1996. (cited on page 164)

176. Schnell, R. Privacy-preserving record linkage. In *Methodological Developments in Data Linkage* (2015), John Wiley & Sons, pp. 201–225. (cited on pages 2 and 205)

177. Schnell, R., Bachteler, T., and Reiher, J. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making 9*, 1 (2009). (cited on pages 2, 4, 5, 8, 17, 22, 23, 30, 37, 40, 44, 53, 65, 69, 70, 71, 72, 74, 79, 82, 84, 85, 93, 95, 98, 100, 101, 126, 129, 134, 144, 153, 167, 178, 184, 186, 190, 200, 207, 208, 209, 213, 214, 215, 216, 217, and 218)

178. Schnell, R., Bachteler, T., and Reiher, J. A novel error-tolerant anonymous linking code. *SSRN Electronic Journal* (January 2011). (cited on pages 23, 24, 48, 69, 74, 153, 160, 178, 190, and 219)

179. Schnell, R., and Borgs, C. Identification of attribute fields in Bloom-filter based record-linkage. *working paper, German Record Linkage Center* (2016). (cited on page 24)

180. Schnell, R., and Borgs, C. Randomized response and balanced Bloom filters for privacy preserving record linkage. In *IEEE International Conference on Data Mining Workshops (ICDMW)* (Barcelona, 2016), pp. 218–224. (cited on pages 23, 24, 73, 74, 76, 132, 134, 153, 160, 161, 167, 200, 217, and 223)

181. Schnell, R., and Borgs, C. XOR-folding for Bloom filter-based encryptions for privacy-preserving record linkage. *SSRN Electronic Journal* (January 2016). (cited on pages 74, 76, 160, 223, and 224)

182. Schnell, R., and Borgs, C. Hardening encrypted patient names against cryptographic attacks using cellular automata. In *IEEE International Conference on Data Mining Workshops (ICDMW)* (Singapore, 2018), pp. 518–522. (cited on pages 160 and 161)

183. Schnell, R., and Borgs, C. Protecting record linkage identifiers using a language model for patient names. *Studies in Health Technology and Informatics 253* (January 2018), 91–95. (cited on page 160)

184. Sehili, Z., and Rahm, E. Speeding up privacy preserving record linkage for metric space similarity measures. *Datenbank-Spektrum 16* (June 2016), 227—-236. (cited on page 43)

185. Singh, S. *The code book: The secret history of codes and code-breaking*. Fourth Estate, 2000. (cited on page 52)

186. Smith, D. Secure pseudonymisation for privacy-preserving probabilistic record linkage. *Journal of Information Security and Applications 34* (2017), 271–279. (cited on pages 25, 37, 44, 85, 95, 98, 100, 101, 163, 164, 167, 178, 184, 186, 190, and 208)

187. Smith, H. J., Milberg, S. J., and Burke, S. J. Information privacy: Measuring individuals' concerns about organizational practices. *Management Information Systems Quarterly* (1996), 167–196. (cited on page 2)

188. Swamidass, S. J., and Baldi, P. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of chemical information and modeling 47*, 3 (2007), 952–964. (cited on page 151)

189. Sweeney, L. Simple demographics often identify people uniquely. *Health (San Francisco) 671* (2000), 1–34. (cited on pages 55 and 56)

190. Sweeney, L. *Computational Disclosure Control: A Primer on Data Privacy Protection*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2001. (cited on pages 43, 45, 80, and 190)

191. Sweeney, L. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS) 10*, 5 (2002), 557–570. (cited on pages 30, 82, 87, and 227)

192. Taylor, L., Zhou, X.-H., and Rise, P. A tutorial in assessing disclosure risk in microdata. *Statistics in Medicine 37*, 25 (2018), 3693–3706. (cited on pages 17, 19, 82, 225, and 226)

193. Templ, M. *Statistical Disclosure Control for Microdata: Methods and Applications in R*. Springer, Cham, 2017. (cited on pages 6 and 82)

194. Ting, K. M. *Precision and Recall*. Springer US, Boston, MA, 2010, pp. 781–781. (cited on page 32)

195. Truta, T., Fotouhi, F., and Barth-Jones, D. Disclosure risk measures for microdata. In *International Conference on Scientific and Statistical Database Management (SSDBM)* (Los Alamitos, CA, USA, July 2003), IEEE, pp. 15–22. (cited on pages 82, 225, and 226)

196. Truta, T. M., and Vinay, B. Privacy protection: p-sensitive k-anonymity property. In *IEEE International Conference on Data Engineering Workshops (ICDEW)* (Atlanta, GA, USA, 2006), pp. 94–94. (cited on page 227)

197. UK Office for National Statistics. Beyond 2011 Matching anonymous data, 2013. Methods and Policies Report M9. (cited on page 78)

198. UK Office for National Statistics. Beyond 2011 Safeguarding data for research: Our policy, 2013. Methods and Policies Report M10. (cited on page 78)

199. Vaiwsri, S., Ranbaduge, T., and Christen, P. Reference values based hardening for Bloom filters based privacy-preserving record linkage. In *Australasian Data Mining Conference (AusDM), CRPIT* (2019), pp. 189–202. (cited on pages 27 and 160)

200. Vatsalan, D. *Scalable and Approximate Privacy-Preserving Record Linkage*. PhD thesis, Research School of Computer Science, The Australian National University, October 2014. (cited on pages 2, 14, 18, 20, 51, 54, 60, 82, 196, and 205)

201. Vatsalan, D., and Christen, P. An iterative two-party protocol for scalable privacy-preserving record linkage. In *Australasian Data Mining Conference (AusDM), CRPIT* (Sydney, Australia, 2012), vol. 134. (cited on pages 2, 19, 26, 27, and 215)

202. Vatsalan, D., and Christen, P. Scalable privacy-preserving record linkage for multiple databases. In *ACM International Conference on Information and Knowledge Management (CIKM)* (Shanghai, 2014). (cited on page 214)

203. Vatsalan, D., and Christen, P. Privacy-preserving matching of similar patients. *Journal of Biomedical Informatics 59* (2016), 285–298. (cited on page 8)

204. Vatsalan, D., Christen, P., O'Keefe, C. M., and Verykios, V. S. An evaluation framework for privacy-preserving record linkage. *Journal of Privacy and Confidentiality (JPC) 6*, 1 (2014), 35–75. (cited on pages 34 and 82)

205. Vatsalan, D., Christen, P., and Rahm, E. Scalable privacy-preserving linking of multiple databases using counting Bloom filters. In *IEEE International Conference on Data Mining Workshops (ICDMW)* (Barcelona, Spain, 2016), pp. 882–889. (cited on pages 20, 29, 54, 153, and 210)

206. Vatsalan, D., Christen, P., and Rahm, E. Scalable multi-database privacy-preserving record linkage using counting Bloom filters. *Clinical Orthopaedics and Related Research (CORR)* (2017). (cited on page 17)

207. Vatsalan, D., Christen, P., and Rahm, E. Incremental clustering techniques for multi-party privacy-preserving record linkage. *Data and Knowledge Engineering 128* (2020), 1206–1223. (cited on page 20)

208. Vatsalan, D., Christen, P., and Verykios, V. S. An efficient two-party protocol for approximate matching in private record linkage. In *Australasian Data Mining Conference (AusDM), CRPIT* (Ballarat, Australia, 2011), vol. 121. (cited on page 15)

209. Vatsalan, D., Christen, P., and Verykios, V. S. A taxonomy of privacy-preserving record linkage techniques. *Information Systems 38*, 6 (2013), 946–969. (cited on pages 15, 16, 18, 22, 32, 81, 98, 167, and 178)

210. Vatsalan, D., Sehili, Z., Christen, P., and Rahm, E. Privacy-preserving record linkage for Big Data: Current approaches and research challenges. In *Handbook of Big Data Technologies*. Springer, 2017, pp. 851–895. (cited on pages 14, 18, 107, and 116)

211. Verykios, V. S., and Christen, P. Privacy-preserving record linkage. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 3*, 5 (2013), 321–332. (cited on page 15)

212. Verykios, V. S., Karakasidis, A., and Mitrogiannis, V. K. Privacy preserving record linkage approaches. *International Journal of Data Mining, Modelling and Management 1*, 2 (2009), 206–221. (cited on pages 16 and 19)

213. Verykios, V. S., Moustakides, G. V., and Elfeky, M. G. A Bayesian decision model for cost optimal record matching. *The International Journal on Very Large Data Bases 12*, 1 (2003), 28–40. (cited on pages 15 and 17)

214. Vidanage, A., Christen, P., Ranbaduge, T., and Schnell, R. A graph matching attack on privacy-preserving record linkage. In *ACM International Conference on Information and Knowledge Management (CIKM)* (October 2020), pp. 1485—-1494. (cited on pages 37, 52, 54, 89, 90, 94, 96, and 102)

215. Vidanage, A., Ranbaduge, T., Christen, P., and Randall, S. A privacy attack on multiple dynamic match-key based privacy-preserving record linkage. *International Journal of Population Data Science (IJPDS) 5*, 1 (April 2020). (cited on pages 37, 50, 54, and 96)

216. Vidanage, A., Ranbaduge, T., Christen, P., and Schnell, R. Efficient pattern mining based cryptanalysis for privacy-preserving record linkage. In *IEEE International Conference on Data Engineering (ICDE)* (Macau, 2019), pp. 1698–1701. (cited on pages 32, 45, 52, 55, 56, 96, 102, 150, 163, and 167)

217. Wang, Q., Xu, Z., and Qu, S. An enhanced k-anonymity model against homogeneity attack. *Journal of Software (JSW) 6* (October 2011), 1945–1952. (cited on pages 82, 87, and 227)

218. Warner, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association (JASA) 60*, 309 (March 1965), 63—66. (cited on pages 28 and 227)

219. Weber, S., Lowe, H., Das, A., and Ferris, T. A simple heuristic for blindfolded record linkage. *Journal of the American Medical Informatics Association (JAMIA) 19* (February 2012), 157–161. (cited on page 15)

220. Wen, Z., and Dong, C. Efficient protocols for private record linkage. In *ACM Symposium on Applied Computing* (Gyeongju, Korea, 2014), pp. 1688–1694. (cited on pages 15, 19, 22, and 31)

221. Wheatley, S., Maillart, T., and Sornette, D. The extreme risk of personal data breaches and the erosion of privacy. *The European Physical Journal B* (January 2016), 7. (cited on page 2)

222. Winkler, I. S. The non-technical threat to computing systems. *Computing Systems 9*, 1 (1996), 3–14. (cited on page 49)

223. Winkler, W. E. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau* (1999), Citeseer. (cited on page 13)

224. Winkler, W. E. Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage. Tech. Rep. RR2000/05, US Bureau of the Census, Washington, DC, 2000. (cited on pages 14, 17, and 221)

225. Winkler, W. E. Overview of record linkage and current research directions. Tech. Rep. RR2006/02, US Bureau of the Census, Washington, DC, 2006. (cited on pages 1 and 14)

226. Winkler, W. E., and Thibaudeau, Y. An application of the Fellegi-Sunter model of record linkage to the 1990 U.S. decennial census. Tech. Rep. RR1991/09, US Bureau of the Census, Washington, DC, 1991. (cited on page 14)

227. Wozniak, P. J. Applied nonparametric statistics (2nd ed.). *Technometrics 33*, 3 (1991), 364–365. (cited on page 110)

228. Yakout, M., Atallah, M. J., and Elmagarmid, A. K. Efficient private record linkage. In *IEEE International Conference on Data Engineering (ICDE)* (Shanghai, March 2009), pp. 1283–1286. (cited on pages 15, 27, and 54)

229. Yao, A. C.-C. How to generate and exchange secrets. In *Annual Symposium on Foundations of Computer Science* (1986), SFCS '86, IEEE, pp. 162–167. (cited on pages 21 and 30)

230. Yeboah-Boateng, E. O., and Amanor, P. M. Phishing, smishing & vishing: An assessment of threats against mobile devices. *Journal of Emerging Trends in Computing and Information Sciences 5*, 4 (2014), 297–307. (cited on page 49)

231. Zhang, B., Cheng, R., and Zhang, F. Secure hamming distance based record linkage with malicious adversaries. *Computers & Electrical Engineering 40*, 6 (2014), 1906–1916. (cited on page 21)