An Efficient Two-Party Protocol for Approximate Matching in Private Record Linkage

Dinusha Vatsalan[1], **Peter Christen**[1], **and Vassilios S. Verykios**[2]

[1] Research School of Computer Science, College of Engineering and Computer Science
The Australian National University, Canberra ACT 0200, Australia
Email: dinusha.vatsalan@anu.edu.au, peter.christen@anu.edu.au

[2] School of Science and Technology
Hellenic Open University, Patras, Greece
Email: verykios@eap.gr

Abstract

The task of linking multiple databases with the aim to identify records that refer to the same entity is occurring increasingly in many application areas. If unique identifiers for the entities are not available in all the databases to be linked, techniques that calculate approximate similarities between records must be used for the identification of matching pairs of records. Often, the records to be linked contain personal information such as names and addresses. In many applications, the exchange of attribute values that contain such personal details between organisations is not allowed due to privacy concerns. The linking of records between databases without revealing the actual attribute values in these records is the research problem known as 'privacy-preserving record linkage' (PPRL). While various approaches have been proposed to deal with privacy within the record linkage process, a viable solution that is well applicable to real-world conditions needs to address the major aspect of scalability of linking very large databases while preserving security and linkage quality.

We propose a novel two-party protocol for PPRL that addresses scalability, security and quality/accuracy. The protocol is based on (1) the use of reference values that are available to both database owners, and allows them to individually calculate the similarities between their attribute values and the reference values; and (2) the binning of these calculated similarity values to allow their secure exchange between the two database owners. Experiments on a real-world database with nearly two million records yield linkage results that have a linear scalability to large databases and high linkage accuracy, allowing for approximate matching in the privacy-preserving context. Since the protocol has a low computational burden and allows quality approximate matching while still preserving the privacy of the databases that are matched, the protocol can be useful for many real-world applications requiring PPRL.

Keywords: Entity resolution, privacy technologies, scalability, approximate matching, similarity measure, binning, two-party protocol.

1 Introduction

In computer science, a long line of research has been conducted in probabilistic record linkage, based on the theoretical foundation provided by Fellegi & Sunter (1969). In today's world many organisations are collecting, storing, processing, analysing and mining fast-growing data sets that contain hundreds or even thousands of millions of records. Analysing such large data sets often requires information from multiple data sources to be aggregated in order to enable more detailed analysis. The process of matching and aggregating records that relate to the same entity from one or more data sets is known as 'record linkage', 'data matching' or 'entity resolution' (Elmagarmid et al. 2007, Herzog et al. 2007). Today, record linkage not only faces computational and operational challenges due to the increasing size of data collections, but also privacy and confidentiality challenges due to growing privacy concerns.

The problem of finding records that represent the same individuals in separate databases without revealing identifying details of these individuals is called the 'privacy-preserving record linkage' (PPRL), 'blind data linkage', or the 'private record linkage' problem (Churches & Christen 2004, Durham et al. 2011, Hall & Fienberg 2010, Verykios et al. 2009).

In the absence of a unique identifier for the entities stored in databases, exact or approximate similarity comparison techniques are applied to the common identifiers (which can contain personal information such as name, address and date of birth) for the identification of matching record pairs (Winkler 2006). Linking records by comparing the encrypted attribute values with a standard cryptographic technique in a three-party protocol seems to be a well-understood solution for PPRL (Churches & Christen 2004, O'Keefe et al. 2004). The attribute values match exactly if the corresponding encrypted values match, and the third party can link records without knowing the actual attribute values. However, a limitation of these methods is that only exact comparisons of values are possible. A small variation in an attribute value results in a completely different encrypted value. In practical applications, the exact matching of identifiers is not always possible due to variations or typographical and other types of errors in real-world data (Hernandez & Stolfo 1995). Applying approximate matching techniques overcomes this problem because these techniques do not rely on exact matches only. Therefore, an approach for approximate matching of identifiers in PPRL is required.

There have been several approaches proposed for approximate matching of identifiers in PPRL (Trepetin 2008). They can be classified into approaches that do or do not require a trusted third party for linkage, which are known as three-party and two-party protocols, respectively (Verykios et al. 2009). The advantages of two-party protocols over three-party protocols are that the former is more secure (because there is no possibility of collusion between two parties), and two-party protocols often have lower communication costs.

A practical PPRL application should address three main factors, which are scalability to large databases, linkage quality, and security. The aim of our paper is to propose a new two-party protocol for approximate matching that is practical in real-world PPRL applications by addressing these three factors. The paper also presents an evaluation of the proposed approach with regard to scalability, quality and security.

In the following section we provide an overview of related work in PPRL. In Section 3, we define the core problem of PPRL which is the focus of this paper, and we propose a scalable two-party solution using binning techniques. Section 4 presents our protocol in detail using small example data sets for illustration, and we analyse the security and the complexity of the protocol. We provide the results of the empirical evaluation of our work using large real-world data sets in Section 5. Finally, we conclude the paper with an outlook to future work in Section 6.

2 Related Work

Various methods for approximate matching in PPRL have been proposed, and several surveys have recently been conducted (Christen 2006*b*, Durham et al. 2011, Karakasidis & Verykios 2010, Trepetin 2008, Verykios et al. 2009). These methods can be classified into those that require a third party for performing the linkage and those that do not.

2.1 Three-party Protocols

A protocol is proposed by Song et al. (2000) that addresses the problem of approximate matching by calculating enciphered permutations for private approximate record matching. However, it is practically impossible to predict all possible permutations in real-world applications. Du et al. (2000) suggested a similar solution for private approximate record matching by pre-computing all possible permutations which are then enciphered. However, such an approach requires an unrealisable amount of storage and is susceptible to known plain text attacks.

A blindfolded multi-party approach is suggested by Churches & Christen (2004) that uses n-gram hash digests to achieve approximate private linkage. All matching hash values are compared using extra information such as the number of n-grams contained in each subset and the number of total n-grams comprising the original value. Though the cost is relatively low, compared to the enciphered permutations and pre-computed scores approaches, this is still a costly approach, because of the power set generation and computation it requires.

The work presented by Al-Lawati et al. (2005) introduces a multi-party secure token blocking protocol for achieving high performance private record linkage by using secure hash signatures for computing secure TF-IDF distances. In their work, three methods are explored which are known as simple blocking, record-aware blocking, and frugal third party blocking. These methods provide a trade-off between privacy and computation and communication cost.

Pang et al. (2009) suggested a protocol based on a set of reference strings common to the database owners. The database owners compute the distances between the reference strings and their attribute values and send the results to a third party that sums these distances and finds the minimum of this sum. If this minimum lies below a threshold the two original attribute values are classified as a match. The performance of the protocol depends crucially on the set of reference strings. In our protocol, we also use reference lists for securely calculating the similarities between attribute values in two databases, which are then compared without sending the actual attribute values to a third party.

A three-party protocol that provides privacy for both data and schema without revealing any information is presented by Scannapieco et al. (2007). This approach transforms records into an embedding metric space while preserving the distance between record values. It is assumed that the third party also holds a global schema to which the schemas of two database owners are mapped. A greedy re-sampling heuristic based on SparseMap is used to map values into a vector space at lower computational costs. However, the experimental results presented by Scannapieco et al. (2007) indicate that the linkage quality is affected by the greedy heuristic re-sampling method.

A hybrid approach that combines anonymisation techniques and cryptographic techniques to solve the private record linkage problem is proposed by Inan et al. (2008). This method uses value generalisation hierarchies in the blocking step, and the record pairs that cannot be blocked are compared in a computationally expensive secure multi-party computation (SMC) step using cryptographic techniques.

Using the one-to-many property of phonetic codes, an approach is proposed by Karakasidis & Verykios (2009) for performing approximate matching in PPRL. The attribute values are encoded using a phonetic encoding algorithm such as Soundex (Christen 2006*a*) and the resulting phonetic codes are mixed with randomly generated phonetic codes and sent to a third party to perform matching. The approach is secure and efficient for approximate matching but is not suitable for linking records based on numerical attributes, since phonetic codes are not suitable for numerical values.

2.2 Two-party Protocols

A two-party protocol is suggested by Atallah et al. (2003) that allows the parties to compute the distance (such as edit-distance) between strings without exchanging them. Due to the large amount of necessary communication required to compute the distance this protocol is unsuited for tasks involving large databases.

Ravikumar et al. (2004) use a secure set intersection protocol which requires extensive computations and is impractical for linking large databases. Yakout et al. (2009) present an approach that is based on transforming the data into vectors as described by Scannapieco et al. (2007) and comparing them without sending them to a third party. Complex numbers are calculated to create a complex plain and in the first step the likely matched pairs are computed by moving an adjustable width slab within the complex plain. The second step computes the actually matching pairs using a scalar product protocol based on randomised vectors.

3 Problem Statement and Proposed Solution

PPRL is the problem of how to identify matching records in different databases more effectively and faster without compromising privacy and security. In practice, the matching of two records can often be determined by similarity functions. Assume *Alice* and *Bob* are the two owners of their respective databases $\mathbf{D}^A$ and $\mathbf{D}^B$. They wish to determine which of their records $R_i^A \in \mathbf{D}^A$ and $R_j^B \in \mathbf{D}^B$ have an overall similarity $sim(R_i^A, R_j^B) \geq s_t$ according to some similarity function sim and minimum similarity threshold s_t. These record pairs are classified as matches. *Alice* and *Bob* agreed to disclose the actual values of some selected attributes of the record pairs that are classified as matches with each other. However, they do not wish to reveal the records that are not matches to each other or to any other party.

Due to the frequency of typographical variations and other errors (Hernandez & Stolfo 1995, Christen 2006*a*), record linkage algorithms have to employ string similarity functions for approximate matching of identifying attribute values such as names and addresses. Similarity functions, such as edit-distance or the Jaro-Winkler algorithm (Christen 2006*a*), are used to calculate the numerical similarity value of two attribute values. The core problem of a PPRL protocol is the calculation of the similarity of two records without revealing the attribute values to any other party.

The use of a public reference table which is common to both database owners has previously been proposed for PPRL in a three-party framework (Pang et al. 2009). Such public reference tables are available to both database owners and can be constructed either with random faked values or from a telephone directory, for example, by extracting all unique names, postcodes, and suburb names. The public reference table is used by the database owners to calculate the similarities between their attribute values and the reference values. These similarities are then sent to a third party that can link the records based on the triangular inequality property of the distance metrics, see Equation 1. Reference values are the values that are known to both database owners *Alice* and *Bob*, while the attribute values are the values that are only known to the corresponding database owner.

$$\begin{aligned} dist(v_i,r) + dist(v_j,r) &\geq dist(v_i,v_j) \\ (1 - sim(v_i,r)) + (1 - sim(v_j,r)) &\geq (1 - sim(v_i,v_j)) \\ 1 - sim(v_i,r) - sim(v_j,r) &\geq -sim(v_i,v_j) \\ sim(v_i,r) + sim(v_j,r) - 1 &\leq sim(v_i,v_j) \qquad (1) \end{aligned}$$

Assume $dist(v_i, v_j)$ is the metric distance between two objects v_i and v_j, and $sim(v_i, v_j) = 1.0 - dist(v_i, v_j)$ is the corresponding similarity between the two objects. Similarity values are assumed to be normalised, such that $0 \leq sim(v_i, v_j) \leq 1$. For an exact match of the two objects the similarity function results in $sim(v_i, v_j) = 1.0$ and for two totally different objects $sim(v_i, v_j) = 0.0$. A distance-based similarity function mainly holds three properties: positivity ($dist(v_i, v_j) \geq 0$), symmetry and triangular inequality. A distance function is symmetric if $dist(v_i, v_j) = dist(v_j, v_i)$. The triangular inequality property states that the direct distance between two objects v_i and v_j is always less than or equal to the combined distance when going through a third object r: $dist(v_i, v_j) \leq dist(v_i, r) + dist(v_i, r)$. Reference values can be used as a third object (r) to calculate the similarity between the actual attribute values (v_i and v_j). Any similarity function that fulfils the conditions of a distance function can be used in this approach.

The similarity between attribute values and reference values ($sim(v_i, r), sim(v_j, r)$) can be calculated by the database owners individually and sent to a third party that can calculate the left hand side (LHS) of Equation 1 by calculating the combined similarity value ($sim(v_i, r) + sim(v_j, r) - 1$). The third party then classifies all the record pairs as matches that have $sim(v_i, r) + sim(v_j, r) - 1 \geq s_t$, where s_t is a threshold value. If the LHS of Equation 1 is greater than s_t, then obviously the right hand side (RHS) of the equation, that is the actual similarity value $sim(v_i, v_j)$ between the two string values v_i and v_j, is also greater than s_t and therefore the pair (v_i,v_j) can be classified as a match. However, the results of an empirical evaluation of this approach conducted by Bachteler et al. (2010) shows inadequate linkage quality in terms of precision and recall. Increasing the size of the reference table improves the linkage quality to some extent but is impractical since this leads to very long run times.

We use the reverse triangular inequality of the distance metric, which is explained by Equation 2, to privately calculate the similarity of two values without exchanging the values.

$$\begin{aligned} |dist(v_i,r) - dist(v_j,r)| &\leq dist(v_i,v_j) \\ |(1 - sim(v_i,r)) - (1 - sim(v_j,r))| &\leq (1 - sim(v_i,v_j)) \\ |-sim(v_i,r) + sim(v_j,r)| &\leq (1 - sim(v_i,v_j)) \\ 1 - |sim(v_j,r) - sim(v_i,r)| &\geq sim(v_i,v_j) \qquad (2) \end{aligned}$$

From the reverse inequality property of the similarity function, we can see that the value for $sim(v_i, v_j)$ (RHS) becomes higher and gets closer to 1.0 if and only if the values for $sim(v_i, r)$ and $sim(v_j, r)$ (LHS) become equal to each other, with r being an object from the reference table. This implies that if the difference between the similarity values of two objects with an object from the reference table is small, then they should be similar to each other.

The scalability factor of the record linkage process can be addressed by blocking the databases using indexing techniques. In large databases comparing all pairs of records is not feasible. The aim of indexing is to reduce this large number of potential comparisons by removing as many pairs of records as possible that correspond to non-matches. A recent survey (Christen 2011) presents detailed reviews of the indexing techniques that can be used in non-privacy-preserving record linkage applications. There have also been several approaches proposed towards this direction within a privacy-preserving setting (Al-Lawati et al. 2005, Inan et al. 2008, Yakout et al. 2009, Inan et al. 2010).

Our protocol indexes the data sets first by blocking the records based on a (phonetic) encoding function such as Soundex (Christen 2006*a*), and then uses public reference lists to generate one or several reference values for each block. These reference values are then used by the database owners to calculate the similarity between their attribute values and the reference values in each block. The similarity of each attribute value in a block is calculated by comparing the value only with the list of reference values that are in it's corresponding block.

Once the similarities are calculated we can perform the linkage by using a third party that links the records based on the triangular inequality of these similarities, as was done by Pang et al. (2009). Since

Table 1: Example Bins of Similarity Range

Bin Label	Start Range	End Range
A	0.5	0.625
B	0.626	0.750
C	0.751	0.875
D	0.876	1.0

Table 2: Matching Bin Combinations (MBC)

Match ID	Attribute 1 (Given Name)	Attribute 2 (Surname)
1	A,B	A,B
2	A,**B**	B,**C**
3	A,**B**	**C**,D
4	B,C	A,B
5	**B**,C	B,**C**
6	**B,C**	**C,D**
7	C,D	A,B
8	C,D	B,C
9	**C**,D	C,**D**

Table 3: Example Records and Reference Values

	Database Owner 1		Database Owner 2	
	Given Name	Surname	Given Name	Surname
Attribute Values	'millar'	'ameile'	'miller'	'amelia'
Reference Values	'myler'	'amalia'	'myler'	'amalia'
Similarity Values	0.7	0.8	0.8	0.9
Bin Labels	B	C	C	D
Match IDs	{1,2,3,4,5,6}	{2,3,5,6,8,9}	{4,5,6,7,8,9}	{3,6,9}
	{2,3,5,6}		{6,9}	

the similarities are pre-calculated this will reduce the run times for linkage and it will be scalable to large databases. This approach provides a scalable three-party solution for approximate matching in a PPRL scenario that can achieve high matching quality. As with other three-party protocols, security is however the major drawback in this three-party protocol. If one of the database owners colludes with the third party they can learn about the other database owner's private data.

Our aim is to develop a two-party protocol by using public reference lists and the reverse of triangular inequality property of distance metrics to measure similarities. If there is a way to exchange the calculated similarities between the database owners without revealing any information, we can simply eliminate the need of a third party for the linkage. Since both database owners know the public reference list values, sending the calculated similarity values can leak some information about the identifiers. We propose a two-party solution for this problem by binning the actual similarity values.

We split the similarity range into a number of bins k ($k > 1$), and each database owner stores the similarities between their attribute values and the reference values as bin labels into which the calculated similarity values fall into. The similarity values have a possible range from 0.0 to 1.0. Since we compare the attribute values only with the reference values that are in their corresponding block, the minimum similarity value will be larger than 0.0, and so we only need to bin similarities in an interval $[s_m, 1.0]$, with $s_m > 0.0$ selected by the user. Binning the similarity range from 0.5 to 1.0 into 4 bins, for example, is shown in Table 1. We will explain this example in detail further below.

We then calculate the Matching Bin Combinations (MBC) based on the binning distance d. The binning distance determines the maximum number of bin differences we allow for each attribute for the approximate matching of attribute values. For example, if

Table 4: Subsets of bin combinations for the $(A, B/C, D)$ combination - Match ID 3 from Table 2

Database owner 1		Database owner 2		Total binning distance (d)
Given name	Surname	Given name	Surname	
A	C	A	C	(0+0) = 0
A	C	A	D	(0+1) = 1
A	C	B	C	(1+0) = 1
A	C	B	D	(1+1) = 2
A	D	A	C	(0+1) = 1
A	D	A	D	(0+0) = 0
A	D	B	C	(1+1) = 2
A	D	B	D	(1+0) = 1
B	C	A	C	(1+0) = 1
B	C	A	D	(1+1) = 2
B	C	B	C	(0+0) = 0
B	C	B	D	(0+1) = 1
B	D	A	C	(1+1) = 2
B	D	A	D	(1+0) = 1
B	D	B	C	(0+1) = 1
B	D	B	D	(0+0) = 0

the binning distance is $d = 1$ for each attribute and we use 2 attributes for the matching (and thus a total binning distance of $d = 2$), the MBC would be the ones that are given in Table 2.

Every candidate for the Matching Bin Combination is given a unique Match ID. Based on the MBC, each database owner calculates the set of Match IDs to which each of the records in their database corresponds to. Then these Match IDs are exchanged. Computing the intersection set of the Match IDs and then exchanging the records that are corresponding to those Match IDs between the database owners provides a two-party solution for our problem.

To illustrate our approach, assume we have two entities in two different databases with the values for the attributes 'Surname' and 'Given Name' as ('millar','ameile') and ('miller','amelia'), as shown in Table 3. Applying the Soundex (Christen 2006*a*) phonetic encoding to these values results in the two blocks 'm460' and 'a540'. Assume that the reference list contains one value for each of these blocks, and they are 'myler' for 'm460' and 'amalia' for 'a540'.

Comparing the attribute values with the corresponding block reference values gives us the similarity values of (0.7,0.8) for ('millar','ameile') and (0.8,0.9) for ('miller','amelia'), which result in the bin combinations (B,C) and (C,D), respectively (see Table 1 for the bin ranges). According to the MBC in Table 2, the corresponding matches would be Match IDs $\{2, 3, 5, 6\}$ and Match IDs $\{6, 9\}$, because the bin combination of B for attribute 'Surname' and C for 'Given Name' appears in Match IDs 2, 3, 5 and 6, whereas the combination of C and D appears in Match IDs 6 and 9 only.

The intersection of these two sets results in set $\{6\}$, which is considered to be a match combination for these two example records, and so the two entities can be classified as a match. If the intersection list is empty the entities do not match.

The MBC calculated here are supersets of all the subsets of bin combinations. For example, if we consider the bin combination of Match ID 3, the subsets of bin combinations would be as shown in Table 4. As shown in this table, if the combination $(A, B/C, D)$ is a match then all subsets of this combination are also matches, since they all have 2 or less than 2 binning distance. This improves the security factor of our approach because there can be many possible matching combinations (16 in this example) for one Match ID.

This parametric solution requires the number of bins k to be determined before the linkage. The selection of k is crucial for the performance of the protocol

Table 5: Notation used in this paper

$\mathbf{D^A}$, $\mathbf{D^B}$	Databases held by database owners Alice and Bob, respectively
$R_i^{\mathbf{A}}$, $R_j^{\mathbf{B}}$	A record in $\mathbf{D^A}$ or $\mathbf{D^B}$, respectively
A, a	Attributes common to $\mathbf{D^A}$ and $\mathbf{D^B}$ that are used for linking, an attribute $a \in A$
v, v_i, v_j	An individual attribute value
r, r_i, r_j	A reference value
$block_a(\cdot)$	Function used to block/index attribute a
b	A blocking key value (BKV): $b = block_a(\cdot)$
c	A compound BKV (CBKV): $c = [block_{a_1}(\cdot), \dots block_{a_{\mid A \mid}}(\cdot)]$
$sim_a(\cdot)$	Function used to calculate similarities between values in attribute a
s_m	Minimum similarity threshold to determine the similarity range $[s_m - 1.0]$
k, d	Number of bins, Maximum number of bin differences to find the matching bin combinations
$enc(\cdot, h)$	Function and key used to hash-encode values
BI, $\mathbf{BI}_a$, **BLI**	Block Index, Block Index for attribute $a \in A$, Block List Index
RLI, $\mathbf{RLI}_a$	Reference List Index for attribute $a \in A$
MBC, **MLI**	Matching Bin Combinations, Match ID List Index

as the three major factors of PPRL protocols, security, scalability and linkage quality, depend on this parameter. The larger the number of bins the smaller the range of each bin is, which results in higher accuracy of the protocol. But the smaller the number of bins the smaller the computational complexity is, as the number of candidates of matching bin combinations is reduced, and the more secure the protocol is due to the higher range of bins. So the number of bins must be carefully chosen. We will experimentally investigate how these three factors are affected by the number of bins in Section 5.

4 A Two-Party Protocol for Scalable and Approximate Secure Matching

In this section we will illustrate the steps (S1 to S9) of our protocol in detail using an example consisting of two small databases with given names and surnames used as the linkage attributes. The notation used throughout the paper is summarised in Table 5.

S1: Alice and Bob agree upon (a) a list of attributes A to be used for the linkage and their priority order such as primary attribute, secondary attribute, etc; (b) one blocking function (phonetic) $block_a(\cdot)$ for each attribute $a \in A$, used to generate blocking key values (BKV) b; (c) a similarity function $sim_a(v, r)$, used to calculate the numerical similarity for a pair of values v and r, where v is an attribute value and r is a reference value, such that for an exact match ($v = r$) $sim_a(v, r) = 1$ and for two totally different values $sim_a(v, r) = 0$; (d) a minimum similarity threshold s_m, which determines the start range of the first similarity bin; (e) the number of bins k to be used; (f) a binning distance d used for finding the candidates of Matching Bin Combinations (MBC) for each attribute; (g) a hash-encoding function $enc(\cdot, h)$ and a corresponding hash key h, used to encode the Compound BKVs (CBKVs), reference lists and finally matching records before they are being exchanged between the database owners. This hash-encoding function can for example be the HMAC (Hashed Message Authentication Code) function (Krawczyk et al. 1997), which encodes a plain-text string into a unique hash-code such that having access to a hash-code only makes it impossible with the current computing techniques to find the plain-text string in a reasonable amount of time. To simplify the illustration we do not apply any hash-encoding function in the example.

S2: Alice and Bob each read their databases (example databases are shown in Figure 1) and independently build their local Block Index (BI) data structures for each linkage attribute, and a Block List Index (BLI) data structure by indexing their databases using the blocking function $block_a(\cdot)$, as is illustrated in Figures 2 and 3. The BI data structures are implemented as an inverted index (Witten et al. 1999). The index keys are the unique encodings of a linkage attribute (the BKVs), and the corresponding lists contain the actual attribute values in a block. The BLI data structure is implemented as a nested inverted index where the keys are the unique encodings of the primary linkage attribute and the values are again inverted indexes with keys being the unique encodings of the secondary linkage attribute and values being the list of unique encodings of the third linkage attribute, for example if the number of linkage attributes is three. The nested inverted indexes for two linkage attributes are shown in Figure 3.

S3: Alice and Bob exchange their BLI data structure with each other. This communication is encrypted, for example using public key encryption (Schneier 1995), such that only Alice and Bob can decrypt each others values. Once the BLI is exchanged, Alice and Bob can generate an intersection list of BLIs, as is illustrated in Figure 3. Exchanging the BLIs to find out the intersection list, which is the list of compound blocks c (individual blocks b for each linkage attribute are grouped to generate the compound block) that are common to both databases, might leak some information about each others data. In order to overcome this, a secure set intersection protocol can be used that enables to find the intersection list of BLI lists securely (Agrawal et al. 2003, Kissner & Song 2005). This is discussed in detail in Section 4.2. Alice and Bob then sort the intersection BLI and find the common individual blocks for each linkage attribute separately, as is illustrated in Figure 4.

S4: The next step is to generate the Reference List Index (RLI) which contains lists of reference values, one for each individual block in the intersection list of BLI. The RLI can be generated by both parties together, for example one could generate reference lists for odd blocks and the other for even blocks, or one for primary attribute blocks and the other for secondary attribute blocks. This is shown in Figure 5. In our example, we assume the number of reference values generated for each block is 1.

S5: Alice and Bob then build their Similarity Index (SI). For each unique individual block b in the intersection BLI, they calculate the similarity of

D^A

RecID	Surname	GivenName
RA1	millar	robert
RA2	myler	amelia
RA3	millar	gail
RA4	peter	robart
RA5	peterra	gail
RA6	smyth	amelie

D^B

RecID	Surname	GivenName
RB1	millar	amelia
RB2	miller	roberto
RB3	petera	gayle
RB4	smitth	amilia
RB5	smeth	amelie
RB6	smeetth	rupert

Figure 1: Example databases held by Alice ($\mathbf{D^A}$) and Bob ($\mathbf{D^B}$) with Surname and Given name attributes, used to illustrate the protocol described in Section 4.

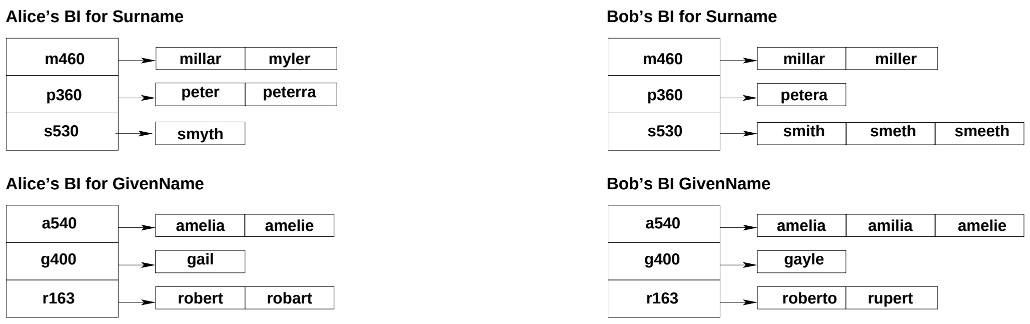

Figure 2: The Block index (BI) of Alice and Bob for the Surname and Given name attributes. The BI is generated in S2 of the protocol as the databases are loaded, and is used in S5 to build the similarity index.

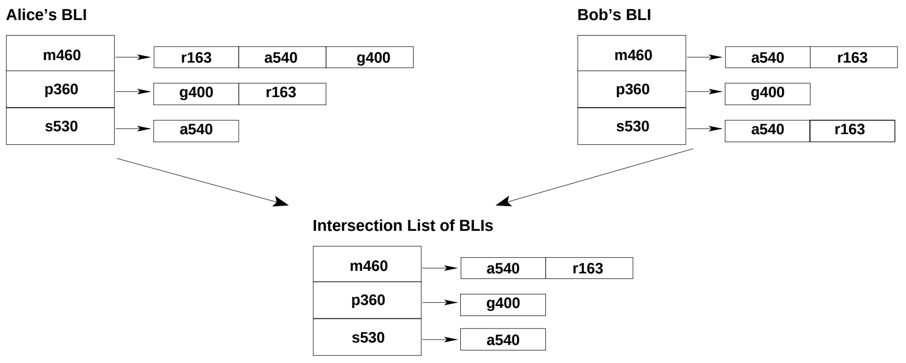

Figure 3: The Block List Index (BLI) of Alice and Bob and the Intersection list of BLIs. Exchanging the BLI in order to calculate the intersection list of the BLIs can reveal some information about the other party's data. This is discussed in detail in Section 4.2. The BLI is generated in S2 of the protocol.

Sorted Compound Blocks

	Surname	Given Name
0	m460	a540
1	m460	r163
2	p360	g400
3	s530	a540

Individual Blocks

Surname	Given Name
m460	a540
p360	r163
s530	g400

Figure 4: The compound blocks c in the sorted intersection list of the BLIs and the individual blocks b for each linkage attribute. The intersection list of BLIs is sorted and the individual blocks are found in S3 of the protocol. The compound blocks are sorted and given index numbers which will be needed in S7 of the protocol.

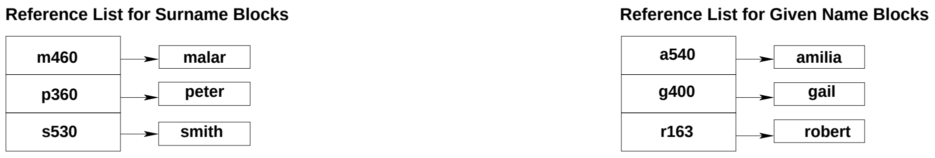
Reference List for Surname Blocks
m460
malar
p360
peter
s530
smith
Reference List for Given Name Blocks
a540
amilia
g400
gail
r163
robert

Figure 5: The reference lists for primary BKVs (Surname attribute) and secondary BKVs (Given name attribute). In the example, we use one reference value per list. These lists are generated in S4 of the protocol.

each unique attribute value in that block (which is stored in their BI as is generated in S2) with the list of reference values of that block, which is retrieved from the RLI. Figure 6 illustrates this for the running example.

S6: In the next step the database owners build the bins with their similarity ranges and the Matching Bin Combinations (MBC), as is illustrated in Figure 7. The bins are stored as an inverted index data structure where the keys are the bin

Alice's SI

Block 'm460' (Surname)

	millar	myler
malar	D (0.8)	C (0.7)

Block 'a540' (GName)

	amelia	amelie
amilia	D (0.8)	C (0.7)

Block 'p360' (Surname)

	peter	peterra
peter	E (1.0)	D (0.8)

Block 'g400' (GName)

	gail
gail	E (1.0)

Block 's530' (Surname)

	smyth
smith	E (0.9)

Block 'r163' (GName)

	robart	robert
robert	E (0.9)	E (1.0)

Bob's SI

Block 'm460' (Surname)

	millar	miller
malar	D (0.8)	C (0.7)

Block 'a540' (GName)

	amelia	amilia	amelie
amilia	D (0.8)	E (1.0)	C (0.7)

Block 'p360' (Surname)

	petera
peter	E (0.9)

Block 'g400' (GName)

	gayle
gail	D (0.8)

Block 's530' (Surname)

	smitth	smeth	smeeth
smith	E (0.9)	E (0.9)	D (0.8)

Block 'r163' (GName)

	roberto	rupert
robert	E (0.9)	D (0.8)

Figure 6: The Similarity Index (SI) which contains the similarities between attribute values and their corresponding reference values calculated using the Jaro-Winkler (Christen 2006*a*) approximate string comparison function, rounded to one digit, along with their corresponding bins. The SI is generated in S5 of the protocol.

Bin Intervals

Label	Range
A	0.5 – 0.59
B	0.6 – 0.69
C	0.7 – 0.79
D	0.8 – 0.89
E	0.9 – 1.0

Matching Bin Combinations

Match ID	Surname	GName
1	A–B	A–B
2	A–B	B–C
3	A–B	C–D
4	A–B	D–E
5	B–C	A–B
6	B–C	B–C
7	B–C	C–D
8	B–C	D–E
9	C–D	A–B
10	C–D	B–C
11	C–D	C–D
12	C–D	D–E
13	D–E	A–B
14	D–E	B–C
15	D–E	C–D
16	D–E	D–E

Figure 7: The bins of similarity and the Matching Bin Combinations (MBC). The bins and their ranges are agreed upon by the database owners in S1 of the protocol. In this example, the number of bins is $k = 5$ and the similarity range is 0.5 to 1.0. The MBC are calculated based on the bins and the binning distance d. In this example, $d = 1$. The bin combinations are generated only for one compound block. Using this, the Match IDs can be calculated using Equation 3 for bin combinations in other blocks as well.

labels/indices and the values are the lists of starting value and ending value of the ranges of each bin. The similarity range between s_m and 1.0 is split into k bins. Based on the bins and the binning distance d, the MBCs are generated with the corresponding Match IDs for each candidate.

S7: Alice and Bob go through their database and build their local Matching Bins of Records (MBR) data structure, as is shown in Figure 8. The MBR data structure contains unique tuples of encoding values of linkage attribute values (CBKVs) and for each unique CBKV, c, it contains an inverted index with keys being the unique tuple of attribute values in that compound block, and values being the lists that contain (a) a list of bin labels for each of the attribute value (which is retrieved from SI, as is generated in S5), (b) a list of Match IDs that correspond to this combination of bin labels (which is retrieved from MBC, as is generated in S6) by using Equation 3, and (c) a list of record IDs that contain this unique tuple of attribute values.

$$\begin{aligned} MatchID &= (compound\_block\_index\_number \\ &\quad \times number\_of\_candidates\_in\_MBC) \\ &\quad + Match\_ID\_in\_MBC \qquad (3) \end{aligned}$$

It is important to note that the MBC data structure is calculated only for one compound block because all the compound blocks will have the same set of candidates of matching bin combinations. The compound blocks in the intersection list of the BLIs is sorted in S3 to find the index numbers of these sorted compound blocks. For example, consider the compound block of $c = [‘p360', ‘g400']$ in Figure 4. It is in the 3rd position in the sorted intersection BLI list. We use a zero-based index in our example, therefore the compound block index number would be 2. The number of candidates in the MBC is 16 in our example. A record with the bin combination of 'D' for the Surname and 'E' for the Given name attribute corresponds to Match IDs 12 and 16 in the MBC (Figure 7). Using Equation 3, the actual Match IDs for this record would be calculated as $(2 \times 16 + 12)$ and $(2 \times 16 + 16)$ which are equal to Match IDs of 44 and 48, the Match IDs for RA5 in Alice's MBR.

S8: Once the MBRs are generated, Alice and Bob retrieve the list of unique Match IDs from their MBR. They then exchange their list of Match IDs with each other and find the intersection of Match IDs, which contains the Match IDs that are common to both database owners. This step is illustrated in Figure 9.

S9: In the final step, as is illustrated in Figures 10 and 11, both Alice and Bob exchange the records (record identifiers) of the matches with each other that are corresponding to the Match IDs in the intersection list of Match IDs. The accumulator is built for storing these matching records, as is shown in Figure 11.

Alice's MBR

BKV Tuple	Surname	GName	Surname Bin	GName Bin	Match IDs	Rec ID
(m460,r163)	millar	robert	D	E	12,16	RA1
(m460,a540)	myler	amelia	C	D	23,24,27,28	RA2
(m460,g400)	millar	gail	D	E	---	RA3
(p360,r163)	peter	robart	E	E	---	RA4
(p360,g400)	peterra	gail	D	E	44,48	RA5
(s530,a540)	smyth	amelie	E	C	63	RA6

Bob's MBR

BKV Tuple	Surname	GName	Surname Bin	GName Bin	Match IDs	Rec ID
(m460,a540)	millar	amelia	D	D	27,31	RB1
(m460,r163)	miller	roberto	C	E	12,16	RB2
(p360,g400)	petera	gayle	E	D	47	RB3
(s530,a540)	smitth	amilia	D	E	60,64	RB4
(s530,a540)	smeth	amelie	D	C	58,59,62,63	RB5
(s530,r163)	smeeth	rupert	D	D	---	RB6

Figure 8: The Matching Bins of Records (MBR) of Alice and Bob. For each of the unique tuple of encoding values (BKVs), it contains the combination of surname and given name attribute values with their corresponding bin labels, a list of Match IDs, and a list of record identifiers that contain the combination. The MBRs are generated in S7 of the protocol. The Match IDs are calculated only for the records that belong to the compound blocks that are in the intersection list of BLIs. The records RA3, RA4 and RB6 in this example belong to the compound blocks of [‘m460’,‘g400’], [‘p360’,‘r163’], and [‘s530’,‘r163’], respectively, which are not in the intersection list of BLIs in Figure 4. In other words, these compound blocks are not common in both databases.

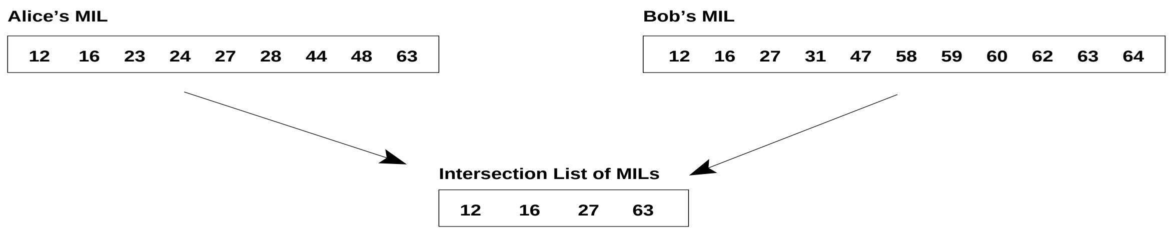

Figure 9: The Match ID List (MIL) of Alice and Bob which contains a list of Match IDs found in their records and the intersection list of MILs. The MILs are generated in S8 of the protocol.

Alice's Matches

Match ID	Rec ID	Surname	GName
12	RA1	millar	robert
16	RA1	millar	robert
27	RA2	myler	amelia
63	RA6	smyth	amelie

Bob's Matches

Match ID	Rec ID	Surname	GName
12	RB2	miller	roberto
16	RB2	miller	roberto
27	RB1	millar	amelia
63	RB5	smeth	amelie

Figure 10: The matches of Alice and Bob for the corresponding Match IDs in the Intersection list of MILs, which are generated and exchanged in S9 of the protocol.

Accumulator

Alice	Bob
RA1	RB2
RA2	RB1
RA6	RB5

Figure 11: The Accumulator generated by Alice and Bob which contains the actual matching record pairs. The accumulator is generated in S9 of the protocol.

4.1 Complexity Analysis

In this section we analyse the computation and communication complexity of our two-party protocol. We assume both databases contain N records, $I = |A|$ attributes that are used for linking the records, and each linkage attribute contains M unique values. Each attribute generates B blocks by applying the $block_a(\cdot)$ functions to index their databases. It is obvious that for large databases it commonly holds that $I \ll B \leq M \ll N$.

In step S1, the agreement of the required functions between Alice and Bob has a constant communication complexity. Reading the databases in step S2 and building the local BI data structures and the BLI data structure requires $O(N)$ of computational complexity if I, B and M are very small compared to N, because building the BI and BLI data structures are $O(I \times M)$ and $O(I \times B)$, respectively.

The exchange of the BLI in step S3 requires the communication of $I \times B$ values for each party, and with I, the number of linkage attributes, being comparatively a very small constant this results in an $O(B)$ communication complexity. Assuming each BLI contains B compound blocks ($I \times B$ individual blocks) calculating the intersection of the two BLIs takes $B \times log(B)$, which results in $O(B\,log(B))$ computation complexity.

We assume the number of reference values used for each individual block in the intersection list of the BLIs is on average R. In step S4, the number of reference values to be generated and exchanged is $I \times B \times R$. With R and I being very small compared to B, this step requires a computation and communication complexity of $O(B)$.

In step S5, assuming each list in the BI that was generated in step S1 contains on average M/B attribute values, each of the $I \times B$ individual blocks requires $(M/B) \times R$ similarity calculations, and thus a total of $I \times M \times R$. Again with I and R being very small, the computation complexity of step S5 is $O(M)$.

Candidates of Matching Bin Combinations are calculated for one compound block based on the number of bins k, the similarity range (which includes the minimum similarity value s_m and the maximum similarity value 1.0) and the binning distance d for each attribute. For each of the candidates a unique Match ID is given. This can be used to calculate the Match IDs for any bin combination in any compound block using Equation 3. The number of candidates is given by $(k-d)^I$ for one compound block and thus the computation complexity is $O((k-d)^I)$.

In step S7, building the MBR by reading the N records requires a total of $O(N)$ computation complexity. In the next step (S8) Alice and Bob exchange the unique Match IDs that are corresponding to the matching combinations found in their records. This is of size $(k-d)^I \times B$, because a maximum of $(k-d)^I$ candidates are calculated for one compound block and each candidate has a corresponding unique Match ID. With the total number of compound blocks being B, this step results in $O((k-d)^I \times B)$ communication complexity. Finding the intersection of these two lists requires $O((k-d)^I B\,log((k-d)^I B))$.

Finally, the generation of the accumulator to store the matches using the Match IDs in the intersection list requires a computation complexity of $O((k-d)^I \times B)$, because a maximum of $(k-d)^I \times B$ Match IDs can be found in the intersection list.

Overall, the communication and computation complexities of our protocol are linear in the size of the databases $O(N)$ and the number of blocks $O(B)$, but is of exponential complexity in the number of attributes I and bins k, $O(k^I)$. The complexity of our protocol greatly depends on the value of k.

4.2 Security Analysis

The protocol assumes that both parties follow the 'honest but curious' behaviour (HBC) (Hall & Fienberg 2010). Both parties are curious, in that they try to find out as much as possible about the other party's inputs while following the protocol. The protocol is secure in the HBC perspective if and only if both parties have no new knowledge at the end of the protocol above what they would have learned from the output of the matched record pairs. We analyse the security of our protocol by discussing what the two parties learn from the data they communicate with each other during the protocol.

There are mainly two steps where we have to consider the security factor in our protocol. One is the exchange of the BLI (that contains compound blocks) which might leak some information regarding the combination of block values in each party's database to other party. Using a secure set intersection (SSI) protocol to find out the intersection set of the compound blocks in each database (without revealing any additional information to either party) will solve this problem. There are two major types of SSI protocols that are commutative encryption (Agrawal et al. 2003) and homomorphic encryption (Kissner & Song 2005).The encryptions of both types of SSI protocols have a linear communication complexity. Since the exchange of the BLI is of $O(B)$ size (see Figure 12), using a SSI protocol for this step is feasible.

The second security issue in our protocol is at the step of exchanging the Match IDs to find the intersection list that contains the Match IDs of matching bin combinations that are common to both databases. This depends on the number of bins. If the number of bins is high then the range of each bin is low and thus the average number of unique attribute values that fall in each bin will be smaller. This results in less overlap in the Matching Bin Combinations (which means some of the candidate Matching Bin Combinations will not be found in any of their records) and allows inferring the combinations of bins in which no records exist when exchanging the Match IDs. If the number of bins is high then the probability of getting bins which do not have any attribute values in them is also high. So the lower the value for the number of bins the higher the security of our protocol.

4.3 Accuracy Analysis

Evaluating the accuracy of our protocol is crucial since we use the bins of similarity values instead of the actual similarity values for the approximate matching of attribute values. In this section we analyse the accuracy of our protocol in terms of the metrics such as precision, recall and f-measure, which are commonly used in Information Retrieval (Raghavan et al. 1989, Manning & Schütze 1999). Based on the classification for true positive (TP), false positive (FP), false negative (FN) and true negative (TN) pairs, the accuracy measures are defined as shown in Equation 4.

$$\begin{aligned} precision &= \frac{\sum TP}{\sum TP + \sum FP} \\ recall &= \frac{\sum TP}{\sum TP + \sum FN} \\ f\!-\!measure &= 2\left(\frac{precision \times recall}{precision + recall}\right) \end{aligned} \tag{4}$$

Accuracy depends on the number of bins. If the number of bins is high then the range of each bin is small which results in more specific ranges of similarity values, and thus the number of FPs and FNs will be less, resulting in higher precision and recall. However, if we increase the number of bins and thus decrease the range of each bin above a certain value then the number of FNs will start to increase, because the number of matches missed as non-matches (FNs) increases. But still the precision is high with an increasing number of bins.

As a result, the f-measure which is the harmonic mean of precision and recall, is expected to increase with the number of bins up-to a certain value. The larger the number of bins used the higher the accuracy of our protocol should be.

5 Experimental Evaluation and Discussion

In this section we present the results of experiments conducted using a real Australian telephone database containing 6,917,514 records. We extracted four attributes commonly used for record linkage: Given name (with 78,336 unique values), Surname (with 404,651 unique values), Suburb (town) name (13,109 unique values), and Postcode (2,632 unique values). These four attributes allowed us to evaluate how the number of attributes used for linkage affects the performance of our protocol. We generated data sets of various sizes by sampling 0.01%, 0.1%, 1%, 10% and 100% of records in the full database twice each in such a way that we obtained pairs of data sets that had an overlap where 25%, 50%, or 75% of records appeared in both the sampled data sets. Table 6 provides an overview of the generated data sets.

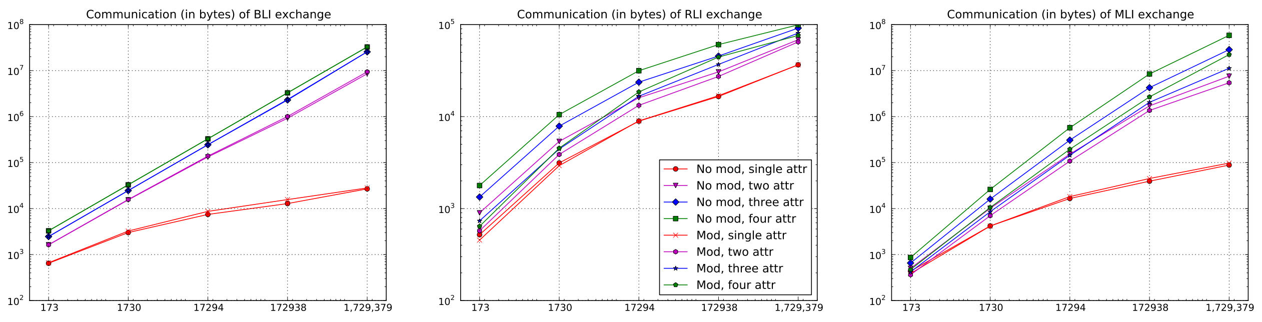

Figure 12: Amount of communication required for different number of linkage attributes, averaged over the results of both database owners over all data set variations as described in Section 5.

Table 6: The number of records in the data sets used for experiments, and the number of records that overlap (i.e. occur in both data sets of a pair). This is considered as the number of true matches.

Data set sizes	25% overlap	50% overlap	75% overlap
173 / 173	38	86	130
1730 / 1730	446	897	1310
17,290 / 17,290	4365	8611	12,973
172,938 / 172,938	42,980	86,363	129,542
1,729,379 / 1,729,379	432,538	86,487	1,297,029

All records that occur in both the data sets are exact matches, which are labelled as 'No mod' (for no modification) in the results figures. To evaluate the performance of approximate matching in the context of 'dirty data' (where attribute values contain typographical errors and variations), we generated another set of data sets (labelled as 'Mod') where we modified two attribute values in each record by applying one randomly selected character edit operation (insert, delete, substitute or transposition).

As encoding functions for blocking the data sets we used Soundex (Christen 2006*a*) for the Given name, Surname and Suburb attributes, while for the Postcode attribute we took the first three digits of the value as the blocking key. The Jaro-Winkler (Christen 2006*a*) string comparison function was used for Given name, Surname, and Suburb name values, while Edit-distance (Christen 2006*a*) was used as a comparison function for Postcode values. The similarity range was set between $s_m = 0.4$ and 1.0 and the binning distance (number of bin differences) for the matching for each linkage attribute was set to $d = 1$ and with 4 attributes to $d = 4$.

We implemented a prototype to evaluate the performance of our protocol using the Python programming language (version 2.7.1). We simulated communication between the parties by creating a directory for each party and writing the communicated data into a file in the receiver's directory. We did not apply any encryption to the communicated data for easy inspection of the files written. All tests were run on a 64-bit Intel Core i7 (2.7 GHz), 8 GBytes of main memory computer running the Ubuntu 11.04 OS platform. The prototype and test data sets are available from the authors.

5.1 Discussion

Figures 12 and 13 shows the scalability of our protocol. Computation complexity is assessed as the total run time required for the linkage, and communication complexity is assessed by the size of the files into which the communicated data are written. All variations of the data sets were used with all the combination of all four attributes. The similarity range was set as $s_m = 0.5$ to 1.0. The value for the number of bins k was set to 5 for these experiments. The results of both the exact and the approximate matching ('No Mod' and 'Mod') are shown in the figures.

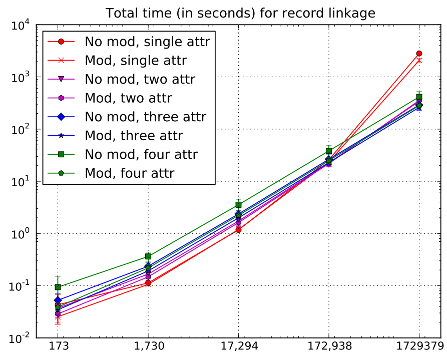

Figure 13: Total run time required for the linkage for different number of linkage attributes, averaged over the results of both database owners over all data set variations as described in Section 5.

As can be seen from Figure 12, the communication complexity of our protocol is linear or sub-linear in the size of the data sets. It increases with the number of attributes I used for linkage. With a smaller number of attributes used, the communication complexity tends to be more sub-linear while with all four attributes used it becomes linear in the size of the data sets.

As expected, the computation complexity of our protocol is linear in the size of the data sets, and it increases with the number of attributes I used for the linkage. Most of the steps in our protocol depend on the number of linkage attributes I and the number of bins k used. However, the linkage performed with one attribute takes longer than with two, three and even four attributes, especially for larger data sets. All the steps performed after the step of calculating the intersection list of the BLIs (step S3) are dependent on this intersection list. With only one linkage attribute there may be many values that exist in both databases and thus the intersection list of the BLIs will be larger than when more than one attribute is used. As a result, the calculation of similarities of these attribute values, the generation of the Matching Bins of Records and building the accumulator takes more time with one attribute only than performing the linkage with several attributes.

Figure 14 presents the experimental results of the complexity, accuracy and security of the protocol for different number of bins k. In these three experiments, all variations of the data sets were used with

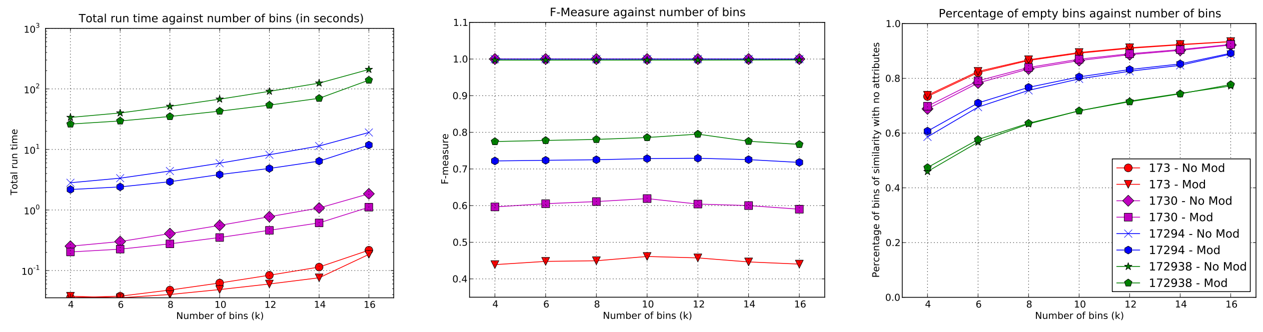

Figure 14: Complexity (left), accuracy (middle), and security (right) of the linkage for different number of bins k, averaged over the results of both database owners over all variations of each data set.

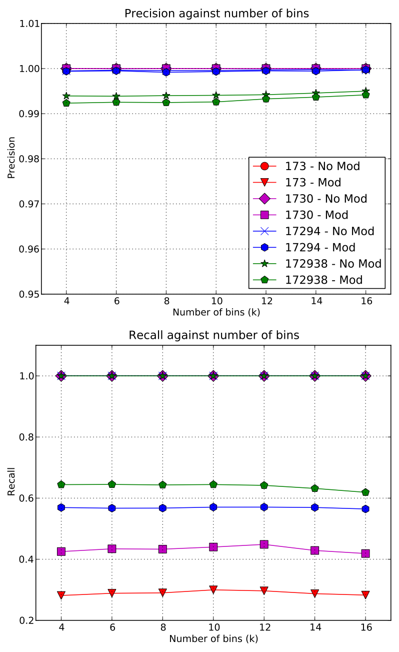

Figure 15: Precision and recall of the linkage, averaged over the results of both database owners over all data set variations as described in Section 5.

'Given name', 'Surname' and 'Postcode' attributes used for the linkage.

The run time is calculated for different number of bin values to evaluate the complexity of the protocol and how it is influenced by the value for k.

Accuracy is measured by running the protocol with different values for the number of bins k and measuring the f-measure as indicated in Equation 4. A pair of records is considered as a true match in our experiments if they have the same record ID. As the results show, our protocol can achieve high accuracy by tuning the parameter k to an optimal value. To analyse the accuracy of our protocol in more detail, we also provide the results of accuracy in terms of precision and recall when k was set to 5 in Figure 15. As discussed in Section 4.3, although the precision increases with k, recall starts decreasing at some point since the number of true matches missed as non-matches starts increasing.

Security is assessed by the percentage of bins that do not have any attribute values when running the protocol with different values for the number of bins, k. As we discussed in Section 4.2, the number of bins used determines the security of our protocol. When exchanging the Match IDs with each other the bin combinations that do not have any records might leak some information. So the percentage of empty bins can be used as a measure to evaluate the security of our protocol. Even with the knowledge of empty bins it is difficult to infer the actual attribute values and also there can be many possible subsets of combinations for a single matching combination (see Table 4).

As we discussed in Sections 4.1 to 4.3, the accuracy and complexity of our protocol increases with the number of bins while the security decreases. This explains that the choice of value for the number of bins, k, is crucial for our protocol.

6 Conclusion

In this paper, we have presented a novel two-party protocol for scalable approximate matching for privacy-preserving record linkage by using reference values and binning the similarity ranges for secure calculation of the similarities between attribute values. Our protocol is linear in the size of the databases to be linked which allows scalability to large databases. This has been validated in our experimental evaluation where we performed the linkage on data sets of up to a size of nearly two million records. However, our protocol is a parametric solution which depends on the number of bins.

As shown in the experimental evaluation the number of bins plays a major role in our protocol in determining the three main factors of the privacy-preserving record linkage protocol, which are security, scalability and accuracy. We aim to tackle this problem of finding the optimal value for the number of bins in our future work. Specifically, we will investigate both analytically and empirically the combinations of security, complexity and accuracy of our protocol with the values for the number of bins.

In our current implementation, we used the Jaro-Winkler string comparison function to measure the similarity between two strings. Another extension to our current work is to compare the performances of the protocol when different approximate string comparison functions are used. We will also investigate how parallelism can improve the performance and security of our protocol. Upon the best determination of the value for the number of bins, our two-party protocol performs well in real-world privacy-preserving record linkage applications.

References

Agrawal, R., Evfimievski, A. & Srikant, R. (2003), Information sharing across private databases, *in* 'ACM SIGMOD', ACM, pp. 86–97.

Al-Lawati, A., Lee, D. & McDaniel, P. (2005), Blocking-aware private record linkage, *in* 'International Workshop on Information Quality in Information Systems', pp. 59–68.

Atallah, M., Kerschbaum, F. & Du, W. (2003), Secure and private sequence comparisons, *in* 'ACM workshop on Privacy in the Electronic Society', pp. 39–44.

Bachteler, T., Schnell, R. & Reiher, J. (2010), An empirical comparison of approaches to approximate string matching in private record linkage, *in* 'Proceedings of Statistics Canada Symposium 2010. Social Statistics: The Interplay among Censuses, Surveys and Administrative Data'.

Christen, P. (2006*a*), A comparison of personal name matching: Techniques and practical issues, *in* 'Workshop on Mining Complex Data, held at IEEE ICDM'06', Hong Kong.

Christen, P. (2006*b*), Privacy-preserving data linkage and geocoding: Current approaches and research directions, *in* 'Workshop on Privacy Aspects of Data Mining, held at IEEE ICDM'06', Hong Kong, pp. 497–501.

Christen, P. (2011), 'A survey of indexing techniques for scalable record linkage and deduplication', *IEEE Transactions on Knowledge and Data Engineering* .

Churches, T. & Christen, P. (2004), 'Some methods for blindfolded record linkage', *BioMed Central Medical Informatics and Decision Making* **4**(9).

Du, W., Atallah, M. & Kerschbaum, F. (2000), Protocols for secure remote database access with approximate matching, *in* 'Proceedings of the 1st ACM Workshop on Security and Privacy in E-Commerce'.

Durham, E., Xue, Y., Kantarcioglu, M. & Malin, B. (2011), 'Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage', *Information Fusion* **In Press**.

Elmagarmid, A. K., Ipeirotis, P. G. & Verykios, V. S. (2007), 'Duplicate record detection: A survey', *IEEE Transactions on Knowledge and Data Engineering* **19**(1), 1–16.

Fellegi, I. P. & Sunter, A. B. (1969), 'A theory for record linkage', *Journal of the American Statistical Society* **64**(328).

Hall, R. & Fienberg, S. (2010), Privacy-preserving record linkage, *in* 'Privacy in Statistical Databases, Springer LNCS 6344', Corfu, Greece, pp. 269–283.

Hernandez, M. A. & Stolfo, S. J. (1995), The merge/purge problem for large databases, *in* 'ACM SIGMOD'95', San Jose.

Herzog, T., Scheuren, F. & Winkler, W. (2007), *Data quality and record linkage techniques*, Springer Verlag.

Inan, A., Kantarcioglu, M., Bertino, E. & Scannapieco, M. (2008), A hybrid approach to private record linkage, *in* 'IEEE ICDE', pp. 496–505.

Inan, A., Kantarcioglu, M., Ghinita, G. & Bertino, E. (2010), Private record matching using differential privacy, *in* 'International Conference on Extending Database Technology'.

Karakasidis, A. & Verykios, V. (2009), Privacy preserving record linkage using phonetic codes, *in* 'Fourth Balkan Conference in Informatics', IEEE, pp. 101–106.

Karakasidis, A. & Verykios, V. (2010), Advances in privacy preserving record linkage, *in* 'E-activity and Innovative Technology, Advances in Applied Intelligence Technologies Book Series', IGI Global, pp. 22–34.

Kissner, L. & Song, D. (2005), Private and threshold set-intersection, Technical report, Carnegie Mellon University.

Krawczyk, H., Bellare, M. & Canetti, R. (1997), HMAC: Keyed-hashing for message authentication, *in* 'Internet RFCs'.

Manning, C. & Schütze, H. (1999), *Foundations of statistical natural language processing*, Vol. 59, MIT Press.

O'Keefe, C., Yung, M., Gu, L. & Baxter, R. (2004), Privacy-preserving data linkage protocols, *in* 'Proceedings of the 2004 ACM workshop on Privacy in the electronic society', pp. 94–102.

Pang, C., Gu, L., Hansen, D. & Maeder, A. (2009), 'Privacy-preserving fuzzy matching using a public reference table', *Intelligent Patient Management* pp. 71–89.

Raghavan, V., Bollmann, P. & Jung, G. (1989), 'A critical investigation of recall and precision as measures of retrieval system performance', *ACM Transactions on Information Systems (TOIS)* **7**(3), 205–229.

Ravikumar, P., Cohen, W. & Fienberg, S. (2004), A secure protocol for computing string distance metrics, *in* 'Workshop on Privacy and Security Aspects of Data Mining held at IEEE ICDM'04', pp. 40–46.

Scannapieco, M., Figotin, I., Bertino, E. & Elmagarmid, A. (2007), Privacy preserving schema and data matching, *in* 'ACM SIGMOD', pp. 653–664.

Schneier, B. (1995), *Applied cryptography: Protocols, algorithms, and source code in C, 2nd Edition*, John Wiley & Sons, Inc., New York.

Song, D., Wagner, D. & Perrig, A. (2000), 'Practical techniques for searches on encrypted data', *Security and Privacy, IEEE Symposium on* p. 44.

Trepetin, S. (2008), 'Privacy-preserving string comparisons in record linkage systems: a review', *Information Security Journal: A Global Perspective* **17**(5), 253–266.

Verykios, V., Karakasidis, A. & Mitrogiannis, V. (2009), 'Privacy preserving record linkage approaches', *Int. J. of Data Mining, Modelling and Management* **1**(2), 206–221.

Winkler, W. E. (2006), Overview of record linkage and current research directions, Technical Report RR2006/02, US Bureau of the Census.

Witten, I. H., Moffat, A. & Bell, T. C. (1999), *Managing Gigabytes, 2nd Edition*, Morgan Kaufmann.

Yakout, M., Atallah, M. & Elmagarmid, A. (2009), Efficient private record linkage, *in* 'IEEE ICDE', pp. 1283–1286.