An Iterative Two-Party Protocol for Scalable Privacy-Preserving Record Linkage

Dinusha Vatsalan and Peter Christen

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

Abstract

Record linkage is the process of identifying which records in different databases refer to the same real-world entities. When personal details of individuals, such as names and addresses, are used to link databases across different organisations, then privacy becomes a major concern. Often it is not permissible to exchange identifying data among organisations. Linking databases in situations where no private or confidential information can be revealed is known as 'privacy-preserving record linkage' (PPRL). We propose a novel protocol for scalable and approximate PPRL based on Bloom filters in a scenario where no third party is available to conduct a linkage.

While two-party protocols are more secure because there is no possibility of collusion between one of the database owners and the third party, these protocols generally require more complex and expensive techniques to ensure that a database owner cannot infer any sensitive information about the other party's data during the linkage process. Our two-party protocol uses an efficient privacy technique called Bloom filters, and conducts an iterative classification of record pairs into matches and non-matches, as selected bits of the Bloom filters are revealed. Experiments conducted on real-world databases that contain nearly two million records, show that our protocol is scalable to large databases while providing sufficient privacy characteristics and achieving high linkage quality.

Keywords: Data matching, entity resolution, privacy, approximate matching, scalability, Bloom filter.

1 Introduction

Privacy-preserving record linkage (PPRL) is the problem of how to efficiently link different databases to identify records that correspond to the same real-world entities without revealing their identities to any party involved in the process, or to any external party or adversary. The three main challenges that a PPRL solution in a real-world context needs to address are (1) scalability to large databases by efficiently conducting the linkage; (2) achieving high quality of the linkage results through the use of approximate (string) matching and effective classification of compared record pairs into matches (two records that are assumed to correspond to the same entity) and non-matches (two records that are assumed to correspond to two different entities); and (3) provision of sufficient privacy guarantees such that the interested parties only learn the records that were reconciled as referring to the same real-world entities (Christen 2012, Clifton et al. 2004, Hall & Fienberg 2010).

One example real-world PPRL application would be where a research team aims to study the correlations between types of car accidents and resulting injuries. Such an analysis requires the linkage of databases from hospitals, health insurance companies and the police (Christen 2012). Another example from the health domain is a health surveillance system that continuously links data from human health data, animal health data, and drugs data to monitor outbreaks of contagious diseases that could lead to epidemics or even pandemics (Clifton et al. 2004). Another application of current interest is where a national security agency needs to collect and link records from a diverse set of databases (such as communication providers, banks, airlines, immigration, and social security) to identify potential terrorism threats (Christen 2006, 2012, Clifton et al. 2004). These example scenarios illustrate that commonly data from different organizations need to be linked, but privacy and confidentiality issues often arise which might prevent such record linkage applications.

Several approaches have been proposed to deal with PPRL over the past two decades (Trepetin 2008, Verykios et al. 2009, Karakasidis & Verykios 2010, Durham et al. 2011, Vatsalan et al. 2013). These approaches can be classified into 'three-party protocols' and 'two-party protocols'. Three-party protocols require a third party for performing the linkage while two-party protocols don't (Christen 2006, 2009, Verykios et al. 2009). The main advantages of two-party protocols over three-party protocols are that they are more secure because there is no possibility of collusion between one of the database owners and the third party, and often they have lower communication costs. However, two-party protocols generally apply more complex techniques, such as Secure Multi-party Computation (SMC) (Clifton et al. 2002, Goldreich 2004, Lindell & Pinkas 2009), to ensure that the two database owners cannot infer any sensitive information from each other during the linkage process. The use of complex techniques, which are computationally intensive, makes PPRL solutions not scalable to large databases and thus are not applicable in real-world contexts.

Among several different privacy techniques that are applied in PPRL solutions, Bloom filters (Bloom 1970) are one efficient technique that can provide adequate privacy guarantees if effectively used. A Bloom filter is a bit string data structure of length l bits,

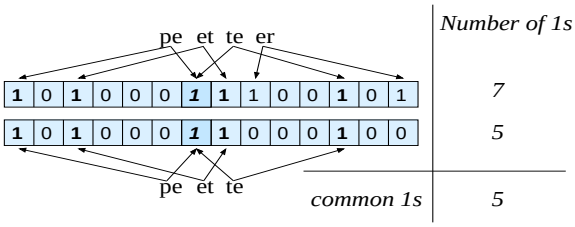

Figure 1: Mapping of strings into Bloom filters and calculating their Dice coefficient similarity.

where all the bits are initially set to 0. k independent hash functions, $h_1, h_2, \ldots, h_k$, each with range $1, \ldots l$, are used to map the elements of a set into the Bloom filter by setting the corresponding bit positions to 1. Bloom filters have previously been used in several three-party or multi-party PPRL solutions.

Schnell et al. (Schnell et al. 2009) were the first to propose a method for approximate matching in PPRL using Bloom filters. In their work, the attribute values of each record in the databases to be linked are concatenated into one string, and the q-grams (sub-strings of length q) of these strings are mapped into Bloom filters using k independent hash functions. These Bloom filters are sent to a third party and the Dice coefficient (Christen 2012) is used to calculate the similarity of two Bloom filters:

$$sim(b^A, b^B) = \frac{2c}{x^A + x^B} \quad (1)$$

where c is the number of common bit positions that are set to 1 in both Bloom filters b^A and b^B (common 1-bits), x^A is the number of bit positions that are set to 1 in b^A, and x^B is the number of bit positions that are set to 1 in b_B. The Dice coefficient is used since it is insensitive to many matching zeros in long Bloom filters (Schnell et al. 2009). For example, mapping the bigrams ($q = 2$) of the two string values 'peter' and 'pete' into $l = 14$ bits long Bloom filters using $k = 2$ hash functions and calculating the Dice coefficient similarity of the two Bloom filters are illustrated in Figure 1.

This approach requires a third party to perform the linkage, since each of the two database owners could mount a dictionary attack on the Bloom filters of the other party because they know the hash functions $h_1 \ldots h_k$ and the length of the Bloom filters l. The approach is efficient because of the use of Bloom filters and it supports approximate matching of values as well, rendering it applicable to real-world conditions. However, as with other three-party protocols, collusion between the parties is a major security drawback of this approach (Schnell et al. 2009). Recent research in PPRL has analysed the weaknesses of Bloom filters in three-party settings using constraint satisfaction cryptanalysis (Kuzu et al. 2011), and novel solutions based on random sampling of bits from attribute level Bloom filters have been proposed (Durham 2012).

Our aim is to develop a two-party protocol for PPRL using Bloom filters. We propose a method that eliminates the need of a third party by iteratively revealing selected bits in the Bloom filters between two database owners, and classifying record pairs into matches and non-matches in an iterative way to reduce the number of record pairs with unknown match status at each iteration without compromising privacy.

Our paper contributes (1) a solution for PPRL in a two-party framework using Bloom filters that is feasible in real-world PPRL applications by addressing the three main challenges that a practical PPRL application poses; (2) an analysis of the proposed solution in terms of complexity, accuracy, and privacy; and (3) an empirical evaluation of the protocol using a large real-world Australian telephone database.

The remainder of the paper is structured as follows. In the following section, we provide an overview of related work in PPRL. In Section 3 we describe the steps of the protocol illustrated with two small sets of Bloom filters. In Sections 4 and 5 we analyse the protocol with regard to performance and privacy, and in Section 6 we validate these analyses through an experimental study. Finally we summarize our findings and discuss future research directions in Section 7.

2 Related Work

Various techniques for privately and efficiently calculating approximate similarities in PPRL have been proposed (Trepetin 2008, Verykios et al. 2009, Karakasidis & Verykios 2010, Durham et al. 2011, Vatsalan et al. 2013). There has been a variety of privacy techniques employed to facilitate PPRL. They include secure hash encoding (Dusserre et al. 1995, Van Eycken et al. 2000, Weber et al. 2012), generalization techniques (Kantarcioglu et al. 2008, Inan et al. 2008, Vatsalan et al. 2011, Mohammed et al. 2011, Karakasidis & Verykios 2012), SMC techniques (Song et al. 2000, Atallah et al. 2003, Ravikumar et al. 2004, Al-Lawati et al. 2005, Inan et al. 2008, 2010, Yakout et al. 2012), differential privacy (Inan et al. 2010), pseudo random functions (Song et al. 2000, O'Keefe et al. 2004, Freedman et al. 2005), Bloom filters (Lai et al. 2006, Schnell et al. 2009, Durham et al. 2010, Karakasidis & Verykios 2011, Durham 2012), reference values (Scannapieco et al. 2007, Pang et al. 2009, Vatsalan et al. 2011, Yakout et al. 2012), phonetic encoding (Karakasidis & Verykios 2011, Karakasidis et al. 2011), and random records (Kargupta et al. 2003, Karakasidis et al. 2011).

Most of the two-party solutions use SMC techniques for the private comparison. Atallah et al. (2003) proposed a two-party protocol where the edit distance algorithm is modified to provide privacy using SMC techniques. Ravikumar et al. (2004) used SMC techniques for the secure computation of several distance functions. The approach of Song et al. (2000) in a two-party context calculates enciphered permutations of values using pseudo random functions and SMC techniques for approximate matching of documents.

Inan et al. (2010) and Yakout et al. (2012) proposed two phase solutions where the first phase is the blocking phase that aims to reduce the number of candidate record pairs by removing pairs that are unlikely to be matches. The remaining candidate pairs are then compared in detail using SMC techniques in the second phase. Inan et al. used differential privacy to partition the perturbed datasets through statistical queries and then generate candidate record pairs from the records in the same partitions. Yakout et al. mapped all the records into a complex plane and then used a slab of a certain width to generate candidate record pairs.

Bloom filters were proposed by Bloom (1970) for efficiently checking set membership. Initially, Bloom filters have been used to support membership queries (Broder et al. 2002). More recently, they have also been used for computing similarities. Several ap-

Table 1: Notation used in this paper.

$\mathbf{D^A}$, $\mathbf{D^B}$	Databases held by database owners Alice and Bob, respectively
$\mathbf{S^A}$, $\mathbf{S^B}$	Lists of values of linkage attributes A for each record in $\mathbf{D^A}$ and $\mathbf{D^B}$, respectively
$b^\mathbf{A}$, $b^\mathbf{B}$	A Bloom filter of each record in $\mathbf{S^A}$ or $\mathbf{S^B}$, respectively
$\mathbf{O^A}$, $\mathbf{O^B}$	Lists of record IDs and number of 1-bits for each record in $\mathbf{D^A}$ and $\mathbf{D^B}$, respectively
$\mathbf{C}$, $\mathbf{C_i}$	List of candidate record pairs, list of candidate record pairs at iteration i
s_t, s_l, s_r	Minimum similarity threshold value to classify a record pair as a match, minimum acceptable similarity threshold value to add random bits, and minimum similarity threshold value to reveal bits in an iteration
l	Length of Bloom filters
$h_1 \dots h_k$, k	Hash functions used to map a set of elements into a Bloom filter, number of hash functions
q	Number of characters that make a q-gram
i	Iteration i, $i > 0$
r, r_i	Number of bit positions revealed, number of bit positions revealed in iteration i
t_i	Total number of bit positions revealed so far up to iteration i, $t_i = \sum_i r_i$
x, x^A, x^B	Number of 1-bits, number of 1-bits in b^A or b^B, respectively
x_i, x_i^A, x_i^B	Total number of 1-bits revealed so far up to iteration i, total number of 1-bits revealed in b^A or b^B so far up to iteration i, respectively
r_{min},r_{max},z_{max}	Minimum number of bits that can be revealed in an iteration, maximum number of total bits to be revealed, maximum number of random bits that can be added
c_{min}, c_i	Minimum number of common 1-bits required in both Bloom filters b^A and b^B, total number of common 1-bits revealed from Bloom filters b^A and b^B so far up to iteration i
d, d_{max}	Difference between x^A and x^B, maximum difference between x^A and x^B to be classified as a 'match'
$sim(\cdot, \cdot)$	Function used to calculate similarities between two Bloom filters b^A and b^B (Dice coefficient)

proaches have been suggested for similarity calculation in PPRL by using Bloom filters (Lai et al. 2006, Schnell et al. 2009, Durham et al. 2010, Karakasidis & Verykios 2011, Durham 2012).

Lai et al. (2006) proposed a multi-party approach that uses Bloom filters for private matching. In their approach, each party partitions its Bloom filters and sends a segment to the other party. The received segments are computed with a logical conjunction (and) and the partial resulting segments are exchanged between the parties. Each party checks its own full Bloom filters with the results and if the membership test is successful then it is considered to be a match. Though the cost of this approach is low since the computation is totally distributed between the parties and the creation and processing of Bloom filters are very fast, the approach is very sensitive to low quality data and is unable to perform approximate matching.

Schnell et al. (2009)'s three-party approach takes into consideration the problem of approximate matching based on a combination of q-grams and Bloom filters as described in Section 1.

Recently, Durham (2012) proposed a three-party framework for PPRL using Bloom filters. In her work, she suggested record level Bloom filter encoding to overcome the problem of cryptanalysis associated with field (or attribute) level encoding (Kuzu et al. 2011), and she used locality-sensitive hash functions for private blocking to reduce the computational complexity. Empirical studies conducted on real datasets show that this approach outperforms existing Bloom filter based approaches.

3 Protocol Description

Two database owners, *Alice* and *Bob*, with databases D^A and D^B, participate in the protocol. We divide the steps of our protocol into three main phases, which are the preparation phase, the length filtering phase, and the iterative classification phase. The notation we use is summarized in Table 1. Figures 2 to 7 illustrate the steps of the protocol.

3.1 Preparation Phase

In the initial preparation phase the database owners prepare their data to be used in the iterative protocol. The steps of this phase are:

1. Alice and Bob agree upon a bit array length l; k hashing functions $h_1 \dots h_k$; the length (in characters) of grams q; the similarity measure $sim(b^A, b^B)$ to measure the similarity of two Bloom filters b^A and b^B; a minimum similarity threshold value s_t, above which two records are classified as a match; the maximum number of bit positions they are willing to reveal to each other r_{max} ($r_{max} \leq l$); and a set of attributes A (linkage attributes) that are used to link the records.
2. Alice and Bob each stores the values of their linkage attributes in a list, $\mathbf{S^A}$ and $\mathbf{S^B}$, respectively, for each of the records in their databases.
3. For every attribute string s in $\mathbf{S^A}$, Alice performs the following steps:
   (a) Alice converts string s into a set of q-grams.
   (b) Alice converts these q-gram sets into a Bloom filter b^A (of that record) of length l using the hash functions $h_1 \dots h_k$. All the attributes of a record are mapped to one single Bloom filter.
4. Alice also counts for each Bloom filter the number of bit positions that are set to 1 (1-bits), x^A, and stores this number along with the identifier of the record into its list $\mathbf{O^A}$, as is illustrated in Figure 2 for the example Bloom filters.
5. For every attribute string s in $\mathbf{S^B}$, Bob performs steps 3 and 4.

3.2 Length Filtering Phase

The second phase of our protocol aims to remove non-matching record pairs using a length filtering method on the Bloom filters. At the end of this phase, candidate record pairs are generated with their corresponding value for the minimum number of common 1-bits they require (c_{min}) to be classified as a match. We use the Dice-coefficient (Equation 1) as the similarity function $sim(\cdot, \cdot)$ to compare two Bloom filters, as it is insensitive to many zeros in Bloom filters. However, any q-gram based similarity function can be used (Christen 2012). Algorithm 1 shows the main steps involved in this phase.

Alice's Bloom Filters

RecID	Bloom Filters	Num 1s (x^A)
RA1	1 1 1 1 0 1 0 1 0 0	6
RA2	0 1 0 0 0 0 1 0 0 1	3
RA3	0 1 1 1 0 1 1 0 0 1	6
RA4	1 1 0 0 1 0 1 1 0 0	5

Bob's Bloom Filters

RecID	Bloom Filters	Num 1s (x^B)
RB1	1 1 1 1 0 1 0 1 1 0	7
RB2	1 1 1 1 0 1 1 0 0 1	7
RB3	1 1 0 1 1 0 1 0 0 0	5

Figure 2: Example Bloom filters held by Alice and Bob for the records in their databases ($\mathbf{D}^A$) and ($\mathbf{D}^B$), respectively, and the number of 1-bits in each of the Bloom filers.

Record Pairs

A	B	x^A	x^B	Length Filter
RA1	RB1	6	7	(|6−7| <= 6/2) Yes
RA1	RB2	6	7	(|6−7| <= 6/2) Yes
RA1	RB3	6	5	(|6−5| <= 6/2) Yes
RA2	RB1	3	7	(|3−7| <= 3/2) No
RA2	RB2	3	7	(|3−7| <= 3/2) No
RA2	RB3	3	5	(|3−5| <= 3/2) No
RA3	RB1	6	7	(|6−7| <= 6/2) Yes
RA3	RB2	6	7	(|6−7| <= 6/2) Yes
RA3	RB3	6	5	(|6−5| <= 5/2) Yes
RA4	RB1	5	7	(|5−7| <= 5/2) Yes
RA4	RB2	5	7	(|5−7| <= 5/2) Yes
RA4	RB3	5	5	(|5−5| <= 5/2) Yes

Candidate Record Pairs

A	B	x^A	x^B	c_{min}
RA1	RB1	6	7	6
RA1	RB2	6	7	6
RA1	RB3	6	5	5
RA3	RB1	6	7	6
RA3	RB2	6	7	6
RA3	RB3	6	5	5
RA4	RB1	5	7	5
RA4	RB2	5	7	5
RA4	RB3	5	5	4

Figure 3: Pruning record pairs that are non-matches (length filtering) according to the number of 1-bits, x^a and x^b, using Equation 2 (left), and candidate record pairs after the length filtering phase, with the minimum number of common 1-bits required to be classified as a match, c_{min}, according to the values of x^A and x^B, calculated using Equation 3 (right). s_t is set to 0.8. The minimum value of all c_{min}, $min(c_{min})$, is 4 which will be used as the value for r_1 in the first iteration ($i = 1$).

Algorithm 1: Length Filtering

Input:
- $\mathbf{O^A}$: List of record IDs and num of 1-bits (r^A, x^A) from Alice
- $\mathbf{O^B}$: List of record IDs and num of 1-bits (r^B, x^B) from Bob
- Minimum similarity threshold s_t

Output:
- List of candidate record pairs with their minimum number of common 1-bits required (c_{min}): $\mathbf{C}$

1: $\mathbf{C} = []$
2: **for** $(r_i^{\mathbf{A}}, x_i^{\mathbf{A}}) \in \mathbf{O^A}$ **do**
3: **for** $(r_i^{\mathbf{B}}, x_i^{\mathbf{B}}) \in \mathbf{O^B}$ **do**
4: $x_{min} = min(x_i^{\mathbf{A}}, x_i^{\mathbf{B}})$
5: $d = |x_i^{\mathbf{A}} - x_i^{\mathbf{B}}|$
6: $d_{max} = \frac{2x_{min}(1-s_t)}{s_t}$
7: **if** $d \leq d_{max}$ **then**
8: $c_{min} = \lfloor \frac{s_t(x_i^{\mathbf{A}} + x_i^{\mathbf{B}})}{2} \rfloor$
9: Append $([r_i^{\mathbf{A}}, x_i^{\mathbf{A}}], [r_i^{\mathbf{B}}, x_i^{\mathbf{B}}], c_{min})$ to $\mathbf{C}$

1. Alice and Bob exchange the number of 1-bits in each of their Bloom filters along with their record identifiers or randomly generated unique ID numbers (lists $\mathbf{O^A}$ and $\mathbf{O^B}$, respectively). They then generate all the record pairs ($|\mathbf{D^A}| \times |\mathbf{D^A}|$ if no blocking function is applied, see Section 4 for how this can be improved) along with the number of 1-bits as is illustrated in Figure 3.

2. The difference between the number of 1-bits in two Bloom filters $d = |x^A - x^B|$, should be less than the maximum bit difference d_{max}, in order to consider the pair as a possible match. Assume $x^A \leq x^B$ and all the bit positions set to 1 in b^A are also set to 1 in b^B ($c = x^A$). This assumption gives the lower bound of the similarity coefficient and the upper bound of bit difference d_{max}. The value for d_{max} can be calculated given the minimum similarity coefficient threshold s_t and number of 1-bits in the Bloom filters, x^A and x^B, as shown in Equation 2.

   All the pairs that have a larger 1-bit difference than d_{max} can be removed without proceeding further since they cannot be matches.

$$
\begin{aligned}
sim(b^A, b^B) &= \frac{2c}{x^A + x^B} \geq s_t \\
\frac{2\min(x^A, x^B)}{\min(x^A, x^B) + (\min(x^A, x^B) + d)} &\geq s_t \\
\frac{2x^A}{x^A + x^A + d} &\geq s_t \\
d &\leq \frac{2x^A(1 - s_t)}{s_t} \\
d_{max} &= \frac{2x^A(1 - s_t)}{s_t}. \qquad (2)
\end{aligned}
$$

   In order to classify a record pair as a match (similarity value above the threshold value s_t), the record pair must have less than or equal to d_{max} number of differences between 1-bits in their Bloom filters. Alice and Bob store only the record pairs that have $|x^A - x^B| \leq d_{max}$, as is illustrated in Figure 3.

   For example, if s_t is set to 0.8, then the difference between 1-bits in two Bloom filters must be at maximum half the value of the smaller value for the 1-bits in the two Bloom filters ($0.5 \times min(x^A, x^B)$) in order to be classified as a match, following $sim(b^A, b^B) \geq 0.8 \Rightarrow \frac{2c}{x_1^A + x_1^B} \geq \frac{8}{10} \Rightarrow \frac{2x_1^A}{x_1^A + (x_1^A + d)} \geq \frac{8}{10} \Rightarrow d \leq 0.5x_1^A$.

3. Alice and Bob now calculate the minimum number of common 1-bits required for a record pair to be classified as a match, c_{min}, for each pair of the remaining candidate records, as is illustrated in Figure 3. This is calculated for each pair using the values for x^A, x^B and s_t as shown in Equation 3, where $\lfloor \cdot \rfloor$ denotes the rounding to the next lowest integer value. The resulting candidate record pairs with the values for x^A, x^B, and c_{min} are stored in the Candidates Index data structure, $\mathbf{C}$, which will be used as an input to

the next phase of the protocol, the iterative classification phase.

$$sim(b^A, b^B) = \frac{2c}{x^A + x^B} \geq s_t$$
$$\frac{2c_{min}}{x^A + x^B} = s_t$$
$$c_{min} = \lfloor \frac{s_t(x^A + x^B)}{2} \rfloor \quad (3)$$

3.3 Iterative Classification Phase

The main task of a record linkage process is the classification of record pairs (Christen 2012). The iterative classification phase is where we classify record pairs into matches, non-matches, and possible matches. This classification needs to be done in such a way that no information about the values that were mapped into Bloom filters is being revealed to the two database owners.

Alice and Bob are prepared to reveal $(l - r_{max})$ bit positions to each other in an iterative way without compromising the sensitive values in their Bloom filters. The number of bits to be revealed in each iteration, r_i, is a crucial parameter to be set as it provides a trade-off between privacy and computational efficiency of the protocol. There are two possible extreme cases.

1. Revealing all the $(l - r_{max})$ bits in one iteration, which is very fast but is not secure since the bit positions are revealed for all the Bloom filter pairs including non-matches as well.

2. Revealing the $(l - r_{max})$ bits in $(l - r_{max})$ iterations where only 1 bit position is revealed in each iteration. This would be the best case for preserving privacy as it removes the non-matches in an iterative way before revealing the rest of the bit positions. This approach is however not scalable to large databases, especially with long Bloom filters, as each iteration requires communication between the database owners.

Hence, a method to reveal an optimal number of bits, r_i, in each iteration is required. We propose a method to calculate this optimal number by finding the smallest value of the minimum number of additional common 1-bits required to classify a pair as a match in each iteration among all the record pairs. The record pair that requires the smallest number of additional common 1-bits among all the other pairs has a security risk if more bit positions are revealed than the minimum number of common 1-bits it requires.

Assume c_i is the total number of common 1-bits revealed so far up to iteration i. The value for $min(c_{min} - c_{i-1})$ $(i > 0)$ is calculated to be used as the value for r_i in the i^{th} iteration. For example, in the first iteration $(i = 1)$, $min(c_{min})$ $(c_0 = 0)$ will be used as the value for the number of bit positions to be revealed, r_1. After r_1 bit positions are revealed in the first iteration, the value for $(c_{min} - c_1)$ will be calculated for each of the remaining record pairs to calculate the value for $r_2 = min(c_{min} - c_1)$ in the second iteration, and then $min(c_{min} - c_2)$ will be used as the value for r_3 in the third iteration, and so on.

The iterative classification phase is done as follows (Algorithm 2 provides an overview of these steps):

1. Among all the $(c_{min} - c_{i-1})$ values for all the unclassified pairs of records, the minimum value, $min(c_{min} - c_{i-1})$, is taken as the lower bound of the number of bits to be revealed in the next iteration. Alice and Bob both will exchange $r_i = min(c_{min} - c_{i-1})$ same bit positions from each of their Bloom filters. For example, if $r_1 = min(c_{min} - c_0) = min(c_{min}) = 4$, then the first 4 bit positions are exchanged in the first iteration, as shown in Figure 4. The total number of bit positions revealed so far up to an iteration i is $t_i = \sum_i r_i$.

   From the exchange of t_i bit positions, three possible cases can occur with each record pair.

   - Case 1 (C1 in Algorithm 2): Record pairs which have c_{min} or more than c_{min} out of t_i bit positions in both Bloom filters (b^A and b^B) set to 1 ($c_i \geq c_{min}$). These pairs are classified as matches.
   - Case 2 (C2 in Algorithm 2): Record pairs which have some or none of the t_i bit positions set to 1 in both Bloom filters b^A and b^B ($c_i < c_{min}$) and the number of additional common 1-bits required ($c_{min} - c_i$) is greater than the number of remaining unrevealed bit positions ($c_i < c_{min}$ and $(c_{min} - c_i) > (l - t_i)$). These pairs are classified as non-matches.
   - Case 3 (C3 in Algorithm 2): Record pairs which have some or none of the t_i bit positions set to 1 in both Bloom filters b^A and b^B ($c_i < c_{min}$) and the number of additional common 1-bits required ($c_{min} - c_i$) is less than or equal to the number of remaining unrevealed bit positions ($c_i < c_{min}$ and $(c_{min} - c_i) \leq (l - t_i)$). These record pairs are classified as possible matches.

Algorithm 2: Iterative Classification Phase

Input:
- **C**: Candidate record pairs from length filtering phase

Output:
- **M**: Set of record pairs classified as matches
- **N**: Set of record pairs classified as non-matches
- **P**: Set of record pairs classified as possible matches

```
1: M = [], N = [], P = C
2: while P ≠ [] do
3:    i = 1, t = 0
4:    while r ≤ r_max do
5:       r_i = min(c_min − c_{i−1})
6:       t = t + r_i
7:       for (b^A, b^B) ∈ P do
8:          x^A = num_1-bits_in_b^A
9:          x^B = num_1-bits_in_b^B
10:         c_min = num_common_1-bits_in_b^A_and_b^B
11:         reveal_bits(r)
12:         x_i^A = total_num_1-bits_revealed_in_b^A
13:         x_i^B = total_num_1-bits_revealed_in_b^B
14:         c_i = total_num_common_1-bits_revealed_in_b^A_and_b^B
15:         if c_i ≥ c_min then      // Case C1
16:            Append (b^A, b^B) to M
17:            Delete (b^A, b^B) from P
18:         else if c_i < c_min and (c_min − c_i) > (l − t) then // C2
19:            Append (b^A, b^B) to N
20:            Delete (b^A, b^B) from P
21:         else if c_i < c_min and (c_min − c_i) ≤ (l − t) then // C3
22:            if ((x^A − x_i^A) < (c_min − c_i)) or
               ((x^B − x_i^B) < (c_min − c_i)) then // C4
23:               Append (b^A, b^B) to N
24:               Delete (b^A, b^B) from P
25:      i = i + 1
26:   for (b^A, b^B) ∈ P do
27:      Do_rehash()
```

Candidate Record Pairs – Iteration 1

A	B	c_{min}	Alice's BF	Bob's BF	$c_{min} - c_1$	$x^A - x^A_1$	$x^B - x^B_1$	Class
RA1(6)	RB1(7)	6	1 1 1 1 ⊠⊠⊠⊠⊠	1 1 1 1 ⊠⊠⊠⊠⊠	2	RA1(2)	RB1(3)	Pos Match
RA1(6)	RB2(7)	6	1 1 1 1 ⊠⊠⊠⊠⊠	1 1 1 1 ⊠⊠⊠⊠⊠	2	RA1(2)	RB2(3)	Pos Match
RA1(6)	RB3(5)	5	1 1 1 1 ⊠⊠⊠⊠⊠	1 1 0 1 ⊠⊠⊠⊠⊠	2	RA1(2)	RB3(2)	Pos Match
RA3(6)	RB1(7)	6	0 1 1 1 ⊠⊠⊠⊠⊠	1 1 1 1 ⊠⊠⊠⊠⊠	3	RA3(3)	RB1(3)	Pos Match
RA3(6)	RB2(7)	6	0 1 1 1 ⊠⊠⊠⊠⊠	1 1 1 1 ⊠⊠⊠⊠⊠	3	RA3(3)	RB2(3)	Pos Match
RA3(6)	RB3(5)	5	0 1 1 1 ⊠⊠⊠⊠⊠	1 1 0 1 ⊠⊠⊠⊠⊠	3	RA3(3)	RB3(2)	*Non Match*
RA4(5)	RB1(7)	5	1 1 0 0 ⊠⊠⊠⊠⊠	1 1 1 1 ⊠⊠⊠⊠⊠	3	RA4(3)	RB1(3)	Pos Match
RA4(5)	RB2(7)	5	1 1 0 0 ⊠⊠⊠⊠⊠	1 1 1 1 ⊠⊠⊠⊠⊠	3	RA4(3)	RB2(3)	Pos Match
RA4(5)	RB3(5)	4	1 1 0 0 ⊠⊠⊠⊠⊠	1 1 0 1 ⊠⊠⊠⊠⊠	2	RA4(3)	RB3(2)	Pos Match

Figure 4: Bloom Filters of Alice and Bob with $t_1 = 4$ ($r_1 = min(c_{min}) = 4$) bits revealed after the first iteration. The calculated values for c_1 are used to calculate the value for r_2 for the next iteration, $r_2 = min(c_{min} - c_1) = 2$.

Candidate Record Pairs – Iteration 2

A	B	$c_{min} - c_1$	Alice's BF	Bob's BF	$c_{min} - c_2$	$x^A - x^A_2$	$x^B - x^B_2$	Class
RA1(2)	RB1(3)	2	1 1 1 1 0 1 ⊠⊠⊠⊠	1 1 1 1 0 1 ⊠⊠⊠⊠	1	RA1(1)	RB1(2)	Pos Match
RA1(2)	RB2(3)	2	1 1 1 1 0 1 ⊠⊠⊠⊠	1 1 1 1 0 1 ⊠⊠⊠⊠	1	RA1(1)	RB2(2)	Pos Match
RA1(2)	RB3(2)	2	1 1 1 1 0 1 ⊠⊠⊠⊠	1 1 0 1 1 0 ⊠⊠⊠⊠	2	RA1(1)	RB2(1)	*Non Match*
RA3(3)	RB1(3)	3	0 1 1 1 0 1 ⊠⊠⊠⊠	1 1 1 1 0 1 ⊠⊠⊠⊠	2	RA3(2)	RB1(2)	Pos Match
RA3(3)	RB2(3)	3	0 1 1 1 0 1 ⊠⊠⊠⊠	1 1 1 1 0 1 ⊠⊠⊠⊠	2	RA3(2)	RB2(2)	Pos Match
RA4(3)	RB1(3)	3	1 1 0 0 1 0 ⊠⊠⊠⊠	1 1 1 1 0 1 ⊠⊠⊠⊠	3	RA4(2)	RB1(2)	*Non Match*
RA4(3)	RB2(3)	3	1 1 0 0 1 0 ⊠⊠⊠⊠	1 1 1 1 0 1 ⊠⊠⊠⊠	3	RA4(2)	RB2(2)	*Non Match*
RA4(3)	RB3(2)	2	1 1 0 0 1 0 ⊠⊠⊠⊠	1 1 0 1 1 0 ⊠⊠⊠⊠	1	RA4(2)	RB3(1)	Pos Match

Figure 5: Bloom Filters of Alice and Bob with $t_2 = 6$ ($r_2 = 2$) bits revealed after the second iteration. The calculated values for c_2 are used to calculate the value for r_3 for the next iteration, $r_3 = min(c_{min} - c_2) = 1$.

Candidate Record Pairs – Iteration 3

A	B	$c_{min} - c_2$	Alice's BF	Bob's BF	$c_{min} - c_3$	$x^A - x^A_3$	$x^B - x^B_3$	Class
RA1(1)	RB1(2)	1	1 1 1 1 0 1 0 ⊠⊠⊠	1 1 1 1 0 1 0 ⊠⊠⊠	1	RA1(1)	RB1(2)	Pos Match
RA1(1)	RB2(2)	1	1 1 1 1 0 1 0 ⊠⊠⊠	1 1 1 1 0 1 1 ⊠⊠⊠	1	RA1(1)	RB2(1)	Pos Match
RA3(2)	RB1(2)	2	0 1 1 1 0 1 1 ⊠⊠⊠	1 1 1 1 0 1 0 ⊠⊠⊠	2	RA3(1)	RB1(2)	Pos Match
RA3(2)	RB2(2)	2	0 1 1 1 0 1 1 ⊠⊠⊠	1 1 1 1 0 1 1 ⊠⊠⊠	1	RA3(1)	RB2(1)	Pos Match
RA4(2)	RB3(1)	1	1 1 0 0 1 0 1 ⊠⊠⊠	1 1 0 1 1 0 1 ⊠⊠⊠	0	RA4(1)	RB3(0)	*Match*

Figure 6: Bloom Filters of Alice and Bob with $t_3 = 7$ ($r_3 = 1$) bits revealed after the third iteration. The calculated values for c_3 are used to calculate the value for r_4 for the next iteration, $r_4 = min(c_{min} - c_3) = 1$.

Candidate Record Pairs – Iteration 4

A	B	$c_{min} - c_3$	Alice's BF	Bob's BF	$c_{min} - c_4$	$x^A - x^A_4$	$x^B - x^B_4$	Class
RA1(1)	RB1(2)	1	1 1 1 1 0 1 0 1 ⊠⊠	1 1 1 1 0 1 0 1 ⊠⊠	0	RA1(0)	RB1(1)	*Match*
RA1(1)	RB2(1)	1	1 1 1 1 0 1 0 1 ⊠⊠	1 1 1 1 0 1 1 0 ⊠⊠	1	RA1(0)	RB2(1)	*Non Match*
RA3(1)	RB1(2)	2	0 1 1 1 0 1 1 0 ⊠⊠	1 1 1 1 0 1 0 1 ⊠⊠	2	RA3(1)	RB1(1)	*Non Match*
RA3(1)	RB2(1)	1	0 1 1 1 0 1 1 0 ⊠⊠	1 1 1 1 0 1 1 0 ⊠⊠	1	RA3(1)	RB2(1)	Pos Match

Figure 7: Bloom Filters of Alice and Bob with $t_4 = 8$ ($r_4 = 1$) bits revealed after the fourth iteration. The pair that is still classified as possible match will need to be re-processed with different hash functions.

2. After having t_i bit positions revealed in iteration i, all the pairs that are classified as matches and non-matches (cases C1 and C2) can be removed from the set of candidate record pairs **C**. Only pairs that are classified as possible matches (case C3) will be taken to the next iteration.

   Based on the revealed bit positions, Alice and Bob calculate the new values for c_i, x_i^A, and x_i^B. Moreover, the values for x_i^A and x_i^B can also be used to prune more non-matches from the pairs of records that were classified as possible matches. Record pairs which have $(c_{min} - c_i) < (x^A - x_i^A)$ or $(c_{min} - c_i) < (x^B - x_i^B)$ can be classified as non-matches and pruned (case C4 in Algorithm 2). For example, if 2 more 1-bits are left unrevealed in b^B ($x^B - x_i^B = 2$) after revealing 4 bit positions in the first iteration, and $(c_{min} - c_i)$ is 3 which means at least 3 more common 1-bits are required for the record pair to be classified as a match from only 2 1-bits in b^B (which is impossible), then this record pair can be removed at this iteration without taking into the next iteration and revealing more bits for this non-matching pair. Record pair RA3 and RB3 in Figure 4 is such a case.

3. For the pairs that are classified as possible matches (case 3), Alice and Bob repeat the steps until r_{max} bit positions are exchanged in an iterative method (r_{max} is set to 8 in our example).

4. The record pairs that are still classified as possible matches in the last step, after r_{max} bit positions have been revealed, need to be re-hashed into new Bloom filters with different hash functions k (lines 26 and 27 in Algorithm 2).

3.4 Computational Complexity

Assuming the number of candidate record pairs is n, the average number of q-grams in each record is Q, the number of hash functions used to map q-grams into a Bloom filter is k, and the length of Bloom filters is l, then the computation cost of this protocol is $O(n*Q*k)$ hash operations and $O((n*l)^2)$ bit comparisons, while the communication cost is $O(n*l)$. The communication complexity of this protocol is therefore linear in the size of the databases.

4 Improving Efficiency

In the length filtering phase, we remove record pairs that have a difference between the number of 1-bits larger than a certain value, depending on the minimum similarity threshold value s_t before starting the iterations as explained in Section 3.2. This reduces the number of candidate record pairs to be processed in the iterative classification phase.

Indexing techniques (Christen 2011) can be applied before performing the linkage based on phonetic encodings (Christen 2012) such that similar records are grouped together. This further reduces the number of candidate record pairs, because only the record pairs that are in the same blocks will be considered as candidate record pairs. Alice and Bob each independently applies an indexing function to their databases and groups records that have the same blocking key value (Christen 2012). The indexing function, for example, can be applied on another set of quasi-identifier attributes or part of the linkage attributes. Secure set intersection protocols (Agrawal et al. 2003, Kissner & Song 2004) can be used to securely identify the list of common blocks in both databases (Vatsalan et al. 2011).

Locality sensitive hashing (LSH) can also be applied to reduce the number of candidate record pairs (Gionis et al. 1999). The LSH method originally addresses the approximate nearest neighbor problem by hashing values such that similar records are put into the same buckets with high probability. Secure set intersection or binning (Vatsalan et al. 2011) can then be used to find the list of common blocks in both databases.

The iterative pruning of candidate record pairs using Bloom filters allows removing pairs that have higher probability of being non-matches before exchanging more bit positions. The aim of our iterative method is to prune the record pairs that are classified as non-matches and matches and thereby reduce the number of pairs of possible matches in each iteration as much as possible. We proposed to reveal $min(c_{min} - c_{i-1})$ number of bits in each iteration. Experiments conducted on a real-world database (see Section 6) show that though many bits are being revealed in the first few iterations, only a very few bits are being revealed in the later iterations which takes many iterations to run and thus makes the process not scalable to large databases.

To overcome this problem, we propose a method for revealing more bits when the number of bits to be revealed becomes very small, without compromising privacy. Assume r_i bits have been revealed in iteration i, among which c_i number of common 1-bits have been found in a record pair which needs $c_{min} - c_i$ more common 1-bits in both Bloom filters in order to classify the pair as a match. If $c_{min} - c_i$ is very small and we still can classify the pair as a match even if no more common 1-bits are found in the later iterations, then this pair will not be at a security risk if more bits are revealed in the next iteration, because it has already been considered as a match. The question now arises what is the maximum value for $c_{non} = c_{min} - c_i$ that can be ignored to classify the pairs as matches without accuracy loss. We introduce another similarity threshold value, s_r, to calculate the value for the minimum number of bits that can be revealed for each pair in an iteration, r_{min}, as shown in Equation 4. This basically expands the calculation of value r_i in step 5 of Algorithm 2 as below. Among the values for c_{non} for all the pairs, the smallest value is taken to be used as the value for the minimum number of bits that can be revealed in all the pairs of Bloom filters in an iteration, $r_{min} = min(c_{non})$.

$$\begin{aligned} s_t - s_r &= \frac{2(c_{non})}{x^A + x^B} \\ r_{min} &= min(c_{non}) \\ r_i &= min(r_i, r_{min}) \end{aligned} \quad (4)$$

If r_i becomes less than r_{min} in an iteration, especially in later iterations, then r_{min} bits will be revealed. It is important to note that the similarity threshold to reveal, s_r, is only used to calculate the value for r_{min} while the similarity threshold s_t is used to classify the pairs. This approach reduces the complexity of the protocol significantly without compromising the privacy of the non-matched record pairs. This is empirically evaluated in Section 6.

5 Privacy Analysis

The amount of privacy provided by this protocol depends on the number of hash functions used (k) and the length of the Bloom filter (l) (Schnell et al. 2009, Kuzu et al. 2011). The values for k and l have to be carefully chosen as these values provide a trade-off between accuracy of the classification and privacy. The higher the value for k/l, the higher the privacy and the lower the accuracy, because the number of q-grams mapped to one single bit increases, which results in less accurate linkage results but makes it harder for an attacker to infer the possible combinations.

Assume the minimum number of bits required to perform a dictionary attack using an external database to infer a bit pattern is t_a. The privacy characteristics provided by our protocol are:

1. More bits are revealed for pairs that are more likely to be matches (Figure 16).
2. Non-matching record pairs are removed in the earlier iterations when only a small number of bits have been revealed ($t_i < t_a$), which therefore cannot be used to infer records using a dictionary attack (Figures 13 and 16).
3. When a sufficient number of bits t_a are revealed for a dictionary attack (iteration i), the remaining unclassified pairs have a minimum similarity that is close enough ($sim_{min}(\mathbf{C}_i) \approx s_t$) to be considered as matches (Figures 13 and 14).

Pruning candidate record pairs that have higher probability of being classified as non-matches at early iterations improves the privacy of the protocol, since the non-matches are removed without revealing more bits in the next iterations. We evaluate the probability of a dictionary attack when different percentage of bits are revealed (see Section 6). We expect the probability to increase with the percentage of bits

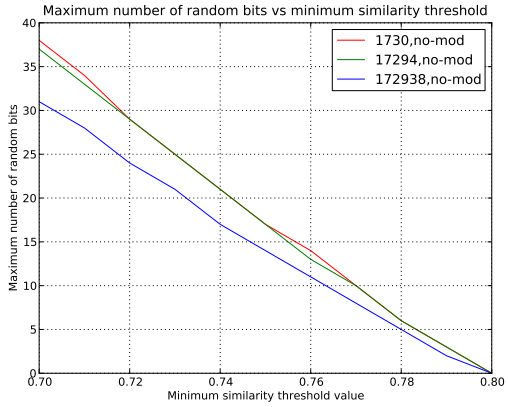

Figure 8: Maximum number of random bits z_{max} that can be added against the minimum lower bound s_l of the similarity threshold value, with $s_t = 0.8$.

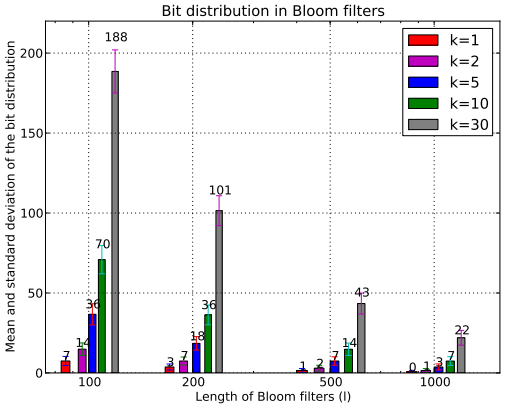

Figure 9: Bit distribution in Bloom filters. Numbers shown are mean values and the error bars are standard deviations.

revealed, as the more bits revealed the smaller the number of possible attribute values one could infer, having access to an external data source (such as a large telephone directory) containing most possible attribute values, which results in a reduced security of the protocol. Hence, only the pairs that have a higher possibility of becoming matches are having a higher probability of a dictionary attack when the percentage of revealed bits increases. Since we hash-map several attribute values from each record into one combined Bloom filter, it is harder for an attacker to infer individual attribute values that correspond to a revealed bit pattern (Durham 2012).

The security parameter which is the maximum number of bits to be revealed in the Bloom filters r_{max} is agreed upon by the two database owners. This determines the privacy of the protocol. A larger value of r_{max} results in less privacy but more record pairs being classified, while a smaller value allows only a smaller number of pairs being classified with higher privacy.

Depending on the data and the distribution of 1-bit patterns, another security issue to be considered with our protocol is that revealing some bits (that have comparatively high sensitive information due to a small number of q-grams that are mapped to those bits) are susceptible to dictionary attacks. We propose two methods for overcoming the problem of revealing the rare bits in Bloom filters that can be attacked with higher probability.

1. **Adding random bits**: Random bits can be added to Bloom filters individually by the database owners in the preparation phase in order to perturb the dataset. The question is how many random bits need to be added to increase the security without compromising accuracy and complexity. When adding random bits three cases can occur. One is when the bits added by the two database owners lead to the same number of additional matching 1-bits at the same positions (common 1-bits) which results in almost the same similarity value. The second case is where some of the added bits are matching and thus the number of additional common 1-bits introduced by the addition of random bits is less than the number of random bits added by the database owners. The third case occurs where the added bits do not match with any bit positions and thus no additional common 1-bits are introduced by adding random bits.

In both the second and third cases, the new similarity value decreases because of the random bits. However, the third case is the worst case and needs to be considered in determining the similarity threshold value. The database owners must agree on a minimum acceptable lower bound of the similarity threshold, s_l, with $s_l < s_t$. If the values for x^A_{min}, x^B_{min}, s_t, and s_l are known, then the maximum number of random bits that can be added by the database owners, z_{max}, can be estimated using Equation 5.

$$\begin{aligned} s_t &= \frac{2 \times c_{min}}{x^A_{min} + x^B_{min}} \\ s_l &= \frac{2 \times c_{min}}{(x^A_{min} + z_{max}) + (x^B_{min} + z_{max})} \\ s_l &= \frac{s_t \times (x^A_{min} + x^B_{min})}{(x^A_{min} + z_{max}) + (x^B_{min} + z_{max})} \\ z_{max} &= \frac{(s_t - s_l) \times (x^A_{min} + x^B_{min})}{2 \times s_l} \end{aligned} \quad (5)$$

Figure 8 shows the maximum number of random bits (z_{max}) that can be individually added to each Bloom filter by the database owners to perturb the bit distribution in Bloom filters against the minimum similarity threshold value that is acceptable without much accuracy loss in the classification results. The maximum number of random bits linearly increases when the minimum similarity threshold decreases.

2. **Simulation attack**: The database owners can individually simulate the protocol and attack their own databases before exchanging the values in order to identify if there exist any bits that map only to a small number of q-grams. Based on that, they can either change the values for k, l, and q, or they can agree on an appropriate value for the security parameter r_{max}. The bit distribution in Bloom filters in a real-world Australian online telephone database with 17,294 records shows that an average of 22 q-grams and a minimum of 14 q-grams are mapped to one single bit when $k = 30$, $q = 2$ and $l = 1000$ (as shown in Figure 9). As can be seen from this figure, the number of q-grams mapped to one bit decreases with l while increasing with k.

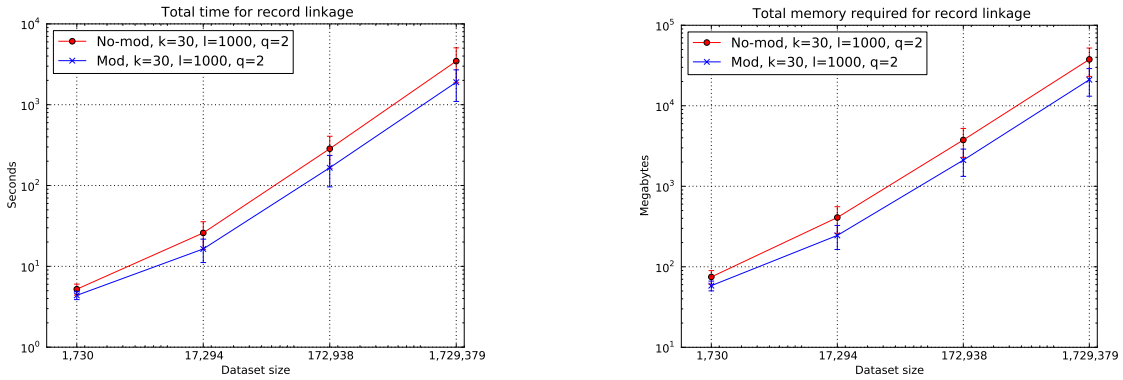

Figure 10: Total run time (left plot) and memory usage (right plot) of the linkage averaged over the results of both database owners over all variations of each dataset. The error bars shown are the standard deviations.

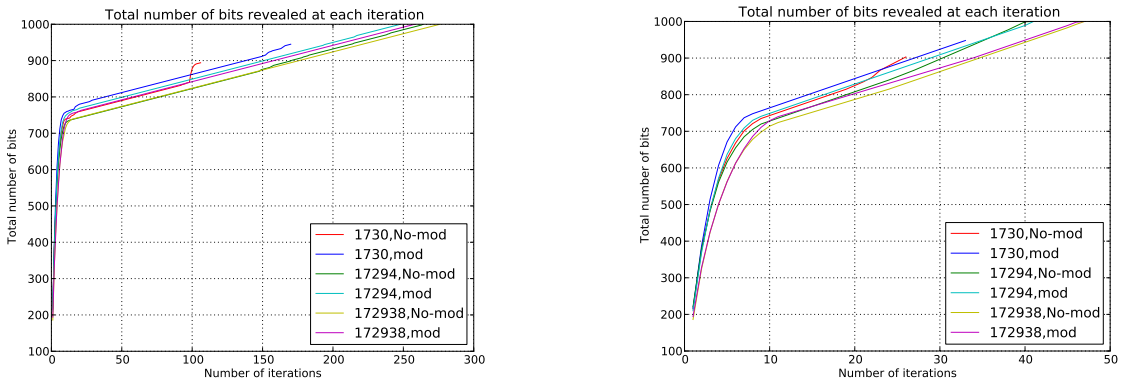

Figure 11: Total number of bits revealed at each iteration without s_r (left plot) and after introducing s_r (right plot). The values for the similarity thresholds were set as $s_t = 0.8$ and $s_r = 0.77$.

Table 2: The number of records in the datasets used for experiments, and the number of records that overlap (i.e. occur in both datasets of a pair). This is considered as the number of true matches.

Dataset sizes	25% overlap	50% overlap	75% overlap
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	864,487	1,297,029

6 Experimental Evaluation

We conducted experiments using a real Australian telephone directory 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). To generate datasets of different sizes, we sampled 0.1%, 1%, 10% and 100% of records in the full database twice each, and stored them into a pair of files such that 25%, 50% or 75% of records appeared in both files of a pair. Table 2 provides an overview of the datasets generated.

The record pairs that occur in both datasets are exact matches (these datasets are labelled as 'No-mod' in the results figures). To investigate the performance of our protocol in the context of 'dirty data' (where attribute values contain errors and variations), we generated another series of datasets (labelled as 'Mod') where we modified each attribute value by applying two randomly selected character edit operations (insert, delete, substitute or transposition). This leads to a much reduced number of exact matching record pairs and allows us to evaluate the accuracy of approximate matching of our protocol.

Following previous work (Schnell et al. 2009), we set the values for the Bloom filter parameters as $l = 1000$, $k = 30$, and $q = 2$. The minimum similarity threshold to classify was set to $s_t = 0.8$ and threshold to reveal bits was set to $s_r = 0.77$. All four attributes were used as the linkage attributes.

We prototyped the protocol using the Python programming language (version 2.7.1). We also implemented an attacker program to evaluate the privacy characteristics of this protocol. We used the attribute values in the full Australian telephone directory as the attacker's reference set of values, and we calculated the probability of a dictionary attack as the number of unique possible values that can be inferred with the bits revealed for every bit pattern in a dataset.

All tests were run on an otherwise idle computer with a 64 bit Intel Xeon (2.4 GHz), 128 GBytes of main memory and running Ubuntu 11.04. The prototype and test datasets are available from the authors.

6.1 Scalability

Figure 10 shows the scalability of our protocol. Computation complexity is assessed as the total run time and memory usage required for the linkage. All variations of the datasets were used. The results of both the exact and the approximate matching ('No-mod' and 'Mod') are shown in the figures. The result figures exhibit a linear complexity trend in the size of the databases which makes the protocol scalable to large databases.

As discussed in Section 4, the number of bits revealed in the later iterations is very small and therefore it takes more iterations to classify the record pairs (see the left plot in Figure 11). With the proposed method of using a second similarity threshold, $s_r = 0.77$, this has been significantly improved, as shown in the right plot in Figure 11. The total number of iterations required to classify all the record pairs is reduced 6-fold (from 300 to 50 iterations for the largest dataset) with the proposed approach using a second threshold $s_r = 0.77$.

The reduction ratio (Christen 2012) of record pairs with unknown match status after classifying record pairs as 'matches' and 'non-matches' at each iteration is shown in Figure 12. As can be seen from the figure, the protocol shows a high increment rate in the reduction ratio after the first few iterations.

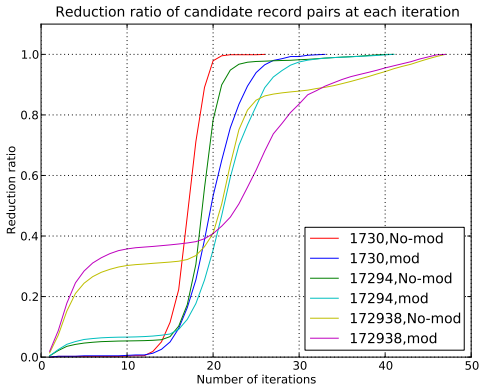

Figure 12: Reduction ratio (Christen 2012) of possible matches at each iteration, using different dataset sizes.

6.2 Privacy

The privacy characteristics of this protocol (as discussed in Section 5) are empirically evaluated in this section assuming that an adversary has access to an external database. The empirical evaluation of probability of a dictionary attack on a dataset consisting of 17,294 records using a public Australian telephone directory as an external database is shown in Figure 13.

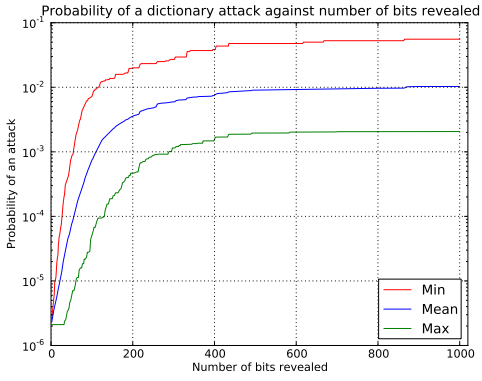

Figure 13: Probability of a dictionary attack from bits revealed (dataset with 17,294 records, reference dataset Australian telephone directory).

This study practically validates that the probability of a dictionary attack increases with the number of bits revealed, and the minimum probability of an attack becomes greater than 0.05 (i.e. the number of values that can be inferred becomes less than 20) only after 800 bits being revealed. When 800 bits are revealed, most of the non-matching record pairs have already been removed (as can be seen from Figure 16), and the minimum similarity value of the remaining record pairs is nearly 0.7 (illustrated in Figure 14), which assures that the privacy of non-matches with similarity below 0.6 is not compromised with this iterative pruning approach.

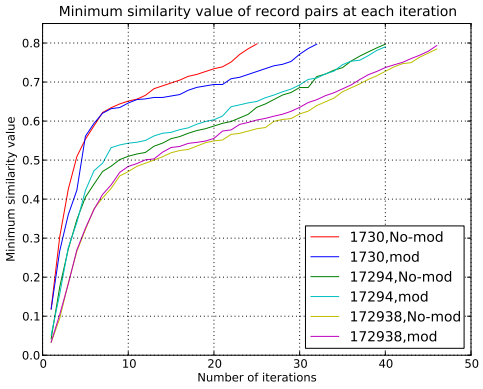

Figure 14: Minimum similarity value of unclassified record pairs at each iteration.

The results of this empirical evaluation of a dictionary attack validates that the privacy of the records corresponding to non-matches is not compromised by this iterative classification approach. The value for security parameter r_{max} can be set after conducting such a simulation attack as described above. For example, in this setting r_{max} can be agreed upon by the database owners to be set as 800.

6.3 Linkage quality

As can be seen from Figure 16, many non-matches are being classified in the first few iterations and then matches are classified more towards the middle to last iterations. The overall reduction ratio (Christen 2012) of candidate record pairs is thus high (Figure 12), while the recall ratio of matches being classified is also high (Figure 15).

The recall ratio is almost 1.0 for the datasets with no modifications ('No-mod'). It is higher (nearly 0.8) with modified datasets as well (a total of 8 edits per record that results in almost 50% modifications in the corresponding q-grams), which explains the aspect of fault-tolerance to data errors by performing approximate matching.

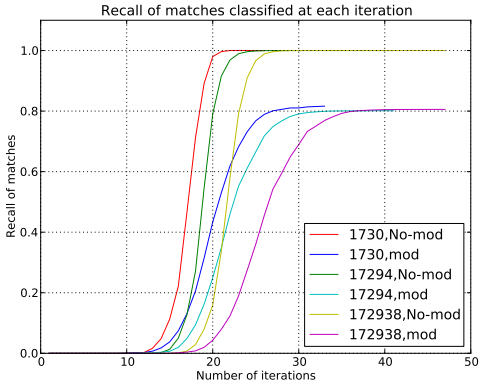

Figure 15: Recall of matches at each iteration, using different dataset sizes.

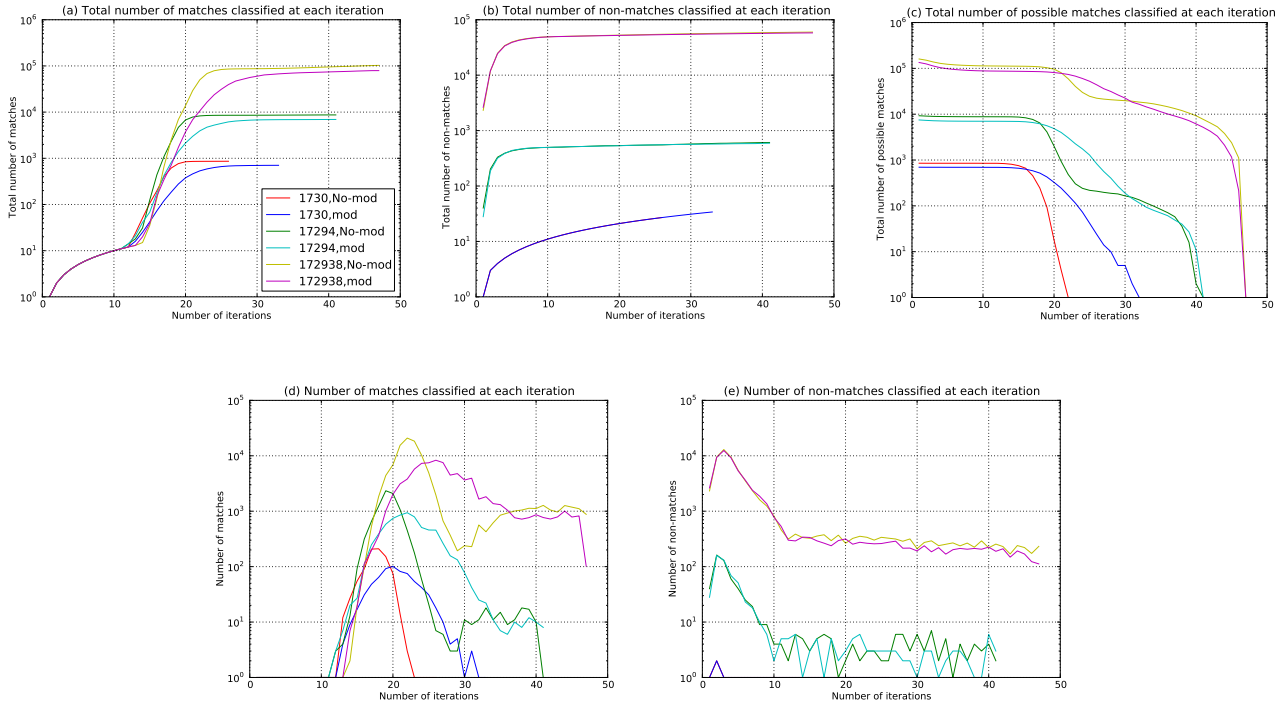

Figure 16: (a) Total number of record pairs classified as matches, (b) non-matches, (c) possible matches, and (d) number of record pairs classified as matches and (e) non-matches at each iteration.

The results of this empirical study validate that this parametric solution performs well by addressing the three main challenges of PPRL which are scalability, privacy, and linkage quality in the current parameter settings. The values for the Bloom filter related parameters l, k, and q play a major role in determining the balancing of these three factors as they provide a trade-off among the three factors.

7 Conclusion

In this paper we proposed a practical two-party protocol for privacy-preserving record linkage by addressing the three main challenges, which are scalability to large databases, high linkage quality results, and sufficient privacy characteristics. With the appropriate determination of values for the parameters, the experimental studies on a real-world database show that our proposed two-party PPRL protocol can perform efficient linkage with high linkage quality while providing adequate privacy characteristics.

In future work, we aim to find the best optimal values for the parameters by theoretically modelling the privacy, accuracy, and complexity of the protocol with different parameter values. Comparing our protocol with other two-party protocols in terms of the three factors is also an interesting research avenue. Another direction would be to study how effectively parallelism can be applied into this protocol.

Learning the values of the parameters such that all three main factors of PPRL are balanced will allow this protocol to be employed in real-world PPRL applications.

References

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

Al-Lawati, A., Lee, D. & McDaniel, P. (2005), Blocking-aware private record linkage, *in* 'Journal of Data and Information Quality', 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.

Bloom, B. (1970), 'Space/time trade-offs in hash coding with allowable errors', *Communications of the ACM* **13**(7), 422–426.

Broder, A., Mitzenmacher, M. & Mitzenmacher, A. (2002), Network applications of Bloom filters: A survey, *in* 'Internet Mathematics'.

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

Christen, P. (2009), Geocode matching and privacy preservation, *in* 'Workshop on Privacy, Security, and Trust in KDD', Springer, pp. 7–24.

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

Christen, P. (2012), *Data Matching*, Data-Centric Systems and Applications, Springer.

Clifton, C., Kantarcioglu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A. & Suciu, D. (2004), Privacy-preserving data integration and sharing, *in* 'ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery', pp. 19–26.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X. & Zhu, M. (2002), 'Tools for privacy preserving distributed data mining', *SIGKDD Explorations* **4**(2), 28–34.

Durham, E. (2012), A framework for accurate, efficient private record linkage, PhD thesis, Vanderbilt University.

Durham, E., Xue, Y., Kantarcioglu, M. & Malin, B. (2010), Private medical record linkage with approximate matching, *in* 'AMIA Annual Symposium Proceedings', p. 182.

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* .

Dusserre, L., Quantin, C. & Bouzelat, H. (1995), 'A one way public key cryptosystem for the linkage of nominal files in epidemiological studies', *Medinfo* **8**, 644–647.

Freedman, M., Ishai, Y., Pinkas, B. & Reingold, O. (2005), 'Keyword search and oblivious pseudorandom functions', *Theory of Cryptography* .

Gionis, A., Indyk, P. & Motwani, R. (1999), Similarity search in high dimensions via hashing, *in* 'Proceedings of the International Conference on Very Large Data Bases', pp. 518–529.

Goldreich, O. (2004), *Foundations of cryptography: Basic applications*, Vol. 2, Cambridge Univ Press.

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

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

Inan, A., Kantarcioglu, M., Ghinita, G. & Bertino, E. (2010), Private record matching using differential privacy, *in* 'EDBT'.

Kantarcioglu, M., Jiang, W. & Malin, B. (2008), A privacy-preserving framework for integrating person-specific databases, *in* 'Privacy in Statistical Databases', Springer, pp. 298–314.

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.

Karakasidis, A. & Verykios, V. (2011), 'Secure blocking+secure matching = secure record linkage', *Journal of Computing Science and Engineering* **5**, 223–235.

Karakasidis, A. & Verykios, V. (2012), Reference table based k-anonymous private blocking, *in* 'ACM Symposium on Applied Computing', Riva del Garda, Italy.

Karakasidis, A., Verykios, V. & Christen, P. (2011), Fake injection strategies for private phonetic matching, *in* 'International Workshop on Data Privacy Management', Leuven, Belgium.

Kargupta, H., Datta, S., Wang, Q. & Sivakumar, K. (2003), On the privacy preserving properties of random data perturbation techniques, *in* 'ICDM', IEEE, pp. 99–106.

Kissner, L. & Song, D. (2004), Private and threshold set-intersection, *in* 'Technical Report', Carnegie Mellon University.

Kuzu, M., Kantarcioglu, M., Durham, E. & Malin, B. (2011), A constraint satisfaction cryptanalysis of Bloom filters in private record linkage, *in* 'Privacy Enhancing Technologies', Springer, pp. 226–245.

Lai, P., Yiu, S., Chow, K., Chong, C. & Hui, L. (2006), An Efficient Bloom filter based Solution for Multiparty Private Matching, *in* 'International Conference on Security and Management'.

Lindell, Y. & Pinkas, B. (2009), 'Secure multiparty computation for privacy-preserving data mining', *Journal of Privacy and Confidentiality* **1**(1), 5.

Mohammed, N., Fung, B. & Debbabi, M. (2011), 'Anonymity meets game theory: secure data integration with malicious participants', *VLDB* **20**(4), 567–588.

O'Keefe, C., Yung, M., Gu, L. & Baxter, R. (2004), Privacy-preserving data linkage protocols, *in* 'ACM Workshop on Privacy in the Electronic Society'.

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

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.

Schnell, R., Bachteler, T. & Reiher, J. (2009), 'Privacy-preserving record linkage using Bloom filters', *BMC Medical Informatics and Decision Making* **9**(1).

Song, D., Wagner, D. & Perrig, A. (2000), 'Practical techniques for searches on encrypted data', *sp* .

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

Van Eycken, E., Haustermans, K., Buntinx, F., Ceuppens, A., Weyler, J., Wauters, E., VAN, O. et al. (2000), 'Evaluation of the encryption procedure and record linkage in the Belgian National Cancer Registry', *Archives of public health* **58**(6), 281–294.

Vatsalan, D., Christen, P. & Verykios, V. (2011), An efficient two-party protocol for approximate matching in private record linkage, *in* 'AusDM, CRPIT 121'.

Vatsalan, D., Christen, P. & Verykios, V. (2013), 'A taxonomy of privacy-preserving record linkage techniques', *to appear in Journal of Information Systems* .

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

Weber, S., Lowe, H., Das, A. & Ferris, T. (2012), 'A simple heuristic for blindfolded record linkage', *Journal of the American Medical Informatics Association* .

Yakout, M., Atallah, M. & Elmagarmid, A. (2012), 'Efficient and practical approach for private record linkage', *Journal of Data and Information Quality* **3**(3), 5.