High Quality and Secure Privacy-Preserving Record Linkage using Random Hashing

Submitted for double-blind review

No Institute Given

Abstract. Record linkage is the task of identifying records that refer to the same entities from several databases. When databases are linked across organizations, privacy and confidentiality leaks become possible, because linkage commonly relies upon sensitive information such as the names, addresses, and other personal details of individuals. Research into privacy-preserving record linkage (PPRL) aims to develop techniques for linking databases such that no private or confidential information has to be revealed from the sensitive source data, and only limited information about matched records is disclosed at the end of the linkage process. In this paper we propose a novel PPRL approach based on random hashing which exhibits both improved privacy characteristics as well as better linkage quality compared to existing PPRL techniques based on Bloom filters. We provide an analysis of the privacy characteristics of our approach and conduct an experimental evaluation on several real-world data sets which validate the superiority of random hashing over Bloom filters.

Keywords: Random maximum margin hashing; Hamming distance; Bloom filter; privacy; data matching; entity resolution.

1 Introduction

Record linkage, also known as entity resolution or data matching [4], is the process of identifying and linking records about the same real-world entity from several databases. This process is widely used in the data preparation phase of data mining projects that require data from various sources to be integrated before they can be analyzed. Linked data can facilitate a large variety of studies that are not feasible on single databases. Traditionally, record linkage has been employed for national censuses and in the health sector, while more recently it has seen application in a wide range of areas including e-commerce, the social sciences, crime and fraud detection, and national security [4].

Record linkage is challenging because it is rare for global unique entity identifiers (like, say social security numbers) to be available in the databases to be linked. Linkage therefore has to rely on private identifying attributes, such as the names and addresses of people. As a result, when databases are linked across organizations, privacy and confidentiality leaks become possible, because linked

databases can reveal highly sensitive information about individuals [22] and the use of personal information is restricted by laws in many countries.

As an example of the benefits of linking, unifying information about individuals in hospital, doctors, police, car, and health insurance databases would allow a public health researcher to identify people who had serious car accidents and study their health following their accidents, thereby identifying unknown patterns that cause serious accidents. Such research could potentially lead to improved road safety measures and help to save many lives [5]. However, having access to all these databases would also reveal criminal histories which should remain private to those without a need to know.

To overcome this privacy challenge, the last two decades have seen research in the emerging area of *privacy-preserving record linkage* (PPRL) [22]. The aim of PPRL is to conduct linkage in such a way that no sensitive data ever needs to be communicated between the parties involved in a linkage, and at the end of a PPRL process all that is revealed is which records refer to the same entity. For the above example, this could be accident and injury details together with basic demographic information of the individuals who had a serious car accident, but not their names, addresses or other identifying details.

A variety of PPRL techniques have been developed over the years, ranging from simple one-way hash encoding, to multi-dimensional embedding and secure multi-party computation (SMC) based techniques. Several recent surveys provide more details on these techniques [3,7,22]. One specific technique that has seen much attention in recent years is based on Bloom filters [2,6,14,16,19], where sensitive attribute values are encoded into bit-strings by the database owners using a secret key only known to them. A major advantage of Bloom filter encoding is that it allows approximate matching of string as well as numerical attribute values [16,20]. Having access to only these Bloom filters and not knowing the secret key makes it possible (though computationally expensive) to re-identify a few frequent attribute values, as has been shown by several recent crypt-analysis attack methods [11,12,17].

However, as the first practical applications of supposedly privacy-preserving record linkage algorithms are being deployed in real-world settings [3,15], it is crucial to ensure these algorithms provide adequate protection from potential privacy-infringing attacks. If even a single sensitive attribute value can potentially be re-identified using such an attack, a PPRL technique might not be acceptable.

Contributions: Our novel PPRL approach aims to overcome the weaknesses of Bloom filter-based encodings to provide significantly improved privacy while still achieving similar quality guarantees in linking compared to Bloom filters. Our approach is based on character q-grams extracted from attribute values, that are then used as features for a random projection hashing approach [8]. The basic idea of the approach is for each database owner to train a set of support vector machines (SVMs) on randomly selected sub-sets of records and with random class labels attached, to then exchange the coefficients of these trained SVMs among the database owners, which then generate one binary vector for each

database record where one SVM contributes one bit of such a vector. These binary vectors are then sent to a third party for comparison and classification. We evaluate this approach on several real-world data sets.

We review related work, and then describe our approach in detail in Sect. 3. We discuss the approach and analyze it with regard to privacy, computational complexity, and linkage quality in Sect 4, and experimentally evaluate it in Sect. 5, before concluding the paper in Sect. 6 with a summary and outlook to future work.

2 Related Work

As recently surveyed [22], techniques for PPRL have been in development since the mid 1990s. The first generation of PPRL techniques allowed for secure exact matching only (using one-way hash functions such as MD5 or SHA), while more recently developed techniques allow for approximate matching of attribute values while also considering scalability to enable PPRL on large databases.

Bloom filters were first proposed by Schnell et al. [16] in 2009 as an efficient and secure encoding method to enable approximate matching of string values in PPRL. Based on this idea, a variety of Bloom filter based encoding methods (including for other data types [20]), hardening techniques, and blocking and matching protocols have been developed [6,14,17,19].

Several crypt-analysis attack methods [11,12,17] have been developed with the aim to re-construct the sensitive values encoded in Bloom filters. The main weaknesses that can be exploited in Bloom filter encoding are that the frequency distributions of Bloom filters, as well as the frequencies of bit patterns within Bloom filters, can be correlated with the frequencies of attribute values [11]. An attacker can obtain such attribute frequency distributions for example from large public databases such as telephone directories. Currently, known Bloom filter encoding techniques are vulnerable to such frequency-based attacks, and no provably secure encoding method for Bloom filters has thus far been developed.

In this paper we aim to develop an alternative approach to Bloom filters that also enables accurate approximate matching for PPRL, but provides much improved privacy compared to Bloom filters.

Our work builds directly on the method proposed by Joly and Buisson [8], whose goal it is to find similar records in large databases by using machine learning methods on the attributes that determine similarity. For example, in a database of D records, suppose that similar values of attributes $a_1, \ldots, a_k$ suggest that a collection of records are similar. Suppose that D is very large. The method consists of creating $n \ll D$ SVMs, each constructed by taking say $m = 1,000$ of the D records, assigning 1/0 labels randomly to them, and then training one SVM on those m records. Then, using each of those n trained SVMs, assign n 0/1 labels to all D records. If two records have similar labels then their values on $a_1, \ldots, a_k$ are likely to be similar. From a privacy point of view, in the absence of the coefficients of the SVMs, the 1/0 vector characterizing a record

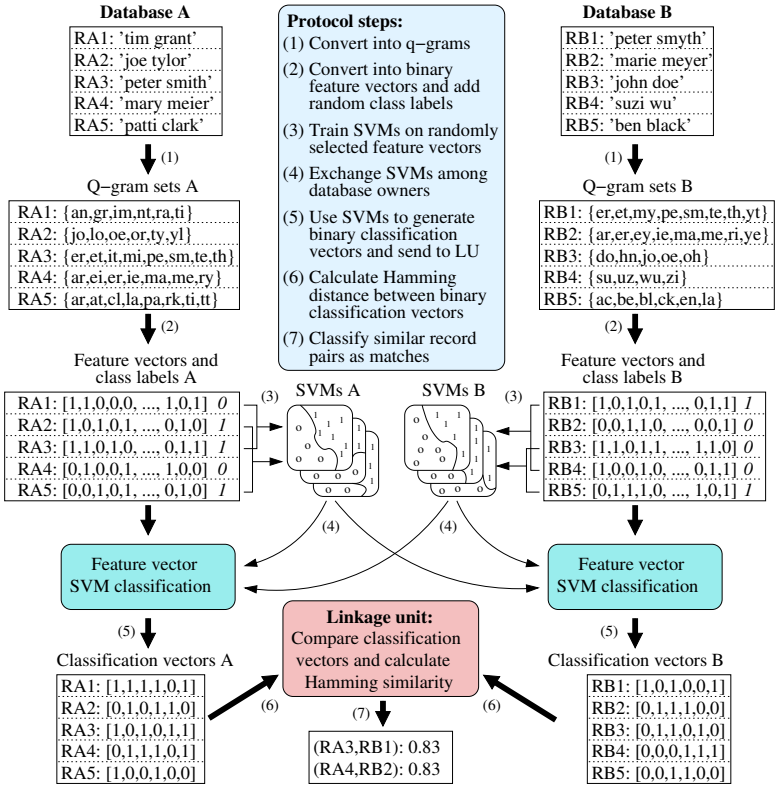

Fig. 1. Outline of our random projection hashing based PPRL protocol as described in Sect. 3. Each database owner (DO) computes support vector machines (SVMs) using random labels on a subset of the records. The DOs only exchange SVM coefficients, and the Linkage Unit (LU) only sees the vectors resulting from applying the joint SVMs on all records in the two databases.

reveals nothing about the $a_1, \ldots, a_k$ attributes of that record. We show later that those vectors are not correlated any more than random 1/0 vectors.

3 Random Projection Hashing based PPRL

We now define the PPRL problem we aim to solve, and then provide a detailed description of our approach to tackle this problem. The steps of our protocol, as described below, are illustrated in Figure 1.

Without loss of generality, we assume two organizations (database owners, DO), named *Alice* and *Bob*, each owning a database, $\mathbf{D}_A$ and $\mathbf{D}_B$, respectively. *Alice* and *Bob* wish to identify records in $\mathbf{D}_A$ and $\mathbf{D}_B$ that have a high similarity on a selected set of common attributes, A, without revealing the actual sensitive values in $\mathbf{D}_A$ and $\mathbf{D}_B$. We formally define the problem of PPRL [22] as:

Definition 1. ***Privacy-preserving record linkage****: Assume* Alice *and* Bob *are the owners of their respective databases* $\mathbf{D}_B$ *and* $\mathbf{D}_B$*. They wish to determine which of their records* $R_i^A \in \mathbf{D}_A$ *and* $R_j^B \in \mathbf{D}_B$ *refer to the same entity according to a classification model* $C(R_i^A, R_j^B)$*.* Alice *and* Bob *are not allowed (or do not wish) to share sensitive values in their* R_i^A *and* R_j^B *with each other or with any other organization. Thus the intended effect of the exchange of data is for* Alice *and* Bob *to learn only which records they have in common according to* C*, and nothing else.*

The model $C(R_i^A, R_j^B)$ classifies record pairs into the class $\mathbf{M}$ of matches (pairs where both records refer to the same entity) and $\mathbf{U}$ of non-matches (pairs where the two records refer to different entities). At the end of the PPRL process, *Alice* and *Bob* may be prepared to disclose to each other, or to an external party (such as a researcher), the actual values of some selected attributes of record pairs in class $\mathbf{M}$ to allow further analysis [22].

To achieve the goal of PPRL, the records in $\mathbf{D}_A$ and $\mathbf{D}_B$ need to be encoded in some form to prevent the actual attribute values in R_i^A and R_j^B to be revealed. Various encoding functions have been proposed in the past [22]. The requirements of such encoding functions are that they must (1) allow approximate matching (due to typographical variations and errors in attribute values), (2) be computationally efficient (to allow matching of very large databases), and (3) not be vulnerable to any attacks by adversaries that aim to identify the sensitive values in $\mathbf{D}_A$ and $\mathbf{D}_B$. We analyze how our approach addresses these three requirements in Sect. 4.

In common with other PPRL protocols, we assume a semi-trusted linkage unit (LU) [16,22], an organization such as a government linkage agency that does not have a database to be linked itself, but which conducts the linkage of the encoded data sent to it by the database owners.

Our novel approach to PPRL applies random projection hashing [8] on the values in attribute(s) A in R_i^A and R_j^B. As will be detailed below, we assume the DOs have agreed upon the set of attributes, A, to use for the linkage, the value of q (length of q-grams), the mapping of q-grams into feature vectors, and the SVM penalty parameter C.

For our approach we employ linear SVMs. In the linear case we can exchange the hyperplane normal $\mathbf{w}$ and the value of b between the two DOs because the SVM classifier can be explicitly expressed as $h(x) = \mathbf{w} \cdot x + b$, where $\mathbf{w}$ is a weighted sum of the support vectors x_i, but we do not need to explicitly exchange these x_is to be able to compute the classifier. In the kernelized case, on the other hand, there is no explicit formulation of $\mathbf{w}$ and the classifier can only be expressed as a function of the support vectors x_is: $h(x) = \sum \mathbf{w}_i \cdot k(x, x_i) + b$ and therefore the x_i need to be exchanged between the DOs. This will reveal to each DO the features (and thus q-grams) of a set of records of the other DO, thus making the approach not secure.

The protocol consists of the following steps (as illustrated in Fig. 1).

1. The two DOs, *Alice* and *Bob*, convert the values in the attributes A in their records $R_i^A \in \mathbf{D}_A$ and $R_j^B \in \mathbf{D}_B$, respectively, into sets of character q-grams

(sub-strings of length q), Q_i^A and Q_j^B (one set per record). This step is the same as done with PPRL techniques based on Bloom filters [6,16,14]. We denote with $\mathbf{Q}_A$ and $\mathbf{Q}_B$ the set of all q-gram sets generated from records in $\mathbf{D}_A$ and $\mathbf{D}_B$, respectively, with $|\mathbf{D}_A| = |\mathbf{Q}_A|$ and $|\mathbf{D}_B| = |\mathbf{Q}_B|$.

2. Each q-gram set $Q_i^A \in \mathbf{Q}^A$ and $Q_j^B \in \mathbf{Q}^B$ is mapped into a binary feature vector B_i^A and B_j^B, respectively, of length $l = |\Sigma|^q$ where Σ is the alphabet of all characters that occur in any value in any attribute in A in any record in $\mathbf{D}_A$ or $\mathbf{D}_B$. Each q-gram is assigned a unique position in these feature vectors, either alphabetically sorted (for example for $q = 2$ and $\Sigma = \{a, \ldots, z\}$: '$aa$' $\rightarrow 0$, 'ab' $\rightarrow 1$, ..., 'zz' $\rightarrow 650$), or using a random permutation agreed upon between (and known only to) the database owners.
3. *Alice* and *Bob* now each train n SVM classifiers in the following way:
   (a) Each DO individually selects n (preferably) non-overlapping training subsets, $\mathbf{T}_k^A$ and $\mathbf{T}_k^B$, $1 \leq k \leq n$, each consisting of m binary feature vectors, B_i^A and B_j^B respectively, they hold, where each could be taken by random sampling without replacement (with a random seed known only to itself). The set of feature vectors for each size m training subset corresponds to m records from $\mathbf{D}_A$ and $\mathbf{D}_B$, respectively.
   The DOs can each select different values for m, with the only constraint that $n \times m$ should be less than the number of records in their database to allow sampling without replacement. Note that each DO keeps this random selection process of training sets secret from any other party (i.e. no other party learns either how a DO has selected its training sets, nor the size of these training sets).
   (b) For each of the feature vectors $B_i^A \in \mathbf{T}_k^A$ and $B_j^B \in \mathbf{T}_k^B$, a class label of 1 or 0 is randomly assigned to that vector with equal probability (i.e. half of the vectors in each $\mathbf{T}_k^A$ and $\mathbf{T}_k^B$ are labeled as being in class 1 and half as being in class 0).
   (c) Each DO then trains n SVMs, one on each of the training sets, $\mathbf{T}_k^A$ and $\mathbf{T}_k^B$, respectively, using the B_i^A and B_j^B and their corresponding 0 and 1 class labels as training data. We denote these n trained SVMs as $\mathbf{S}_k^A$ and $\mathbf{S}_k^B$, respectively, with $1 \leq k \leq n$.
4. The two DOs now exchange their trained SVMs, $\mathbf{S}_k^A$ and $\mathbf{S}_k^B$, with each other (specifically, their coefficients), and each generates two lists of n SVMs, one list with its own SVMs and the other list with the other DO's SVMs.
5. Each database owner now independently applies each of these $2 \times n$ SVMs on all of its own records $R_i^A \in \mathbf{D}_A$ and $R_j^B \in \mathbf{D}_B$, respectively, as converted into binary feature vectors B_i^A and B_j^B in step 2. The two resulting binary vectors, C_i^A and C_j^B, of length n for each record are then combined into a new vector, C_i^X, by applying a bit-wise XOR operation: $C_i^X[b] = C_i^A[b] \oplus C_i^B[b] = (C_i^A[b] \vee C_i^B[b]) \wedge \neg(C_i^A[b] \wedge C_i^B[b])$, $1 \leq b \leq n$. Each position in such a vector is the combined binary classification outcome of applying one specific SVM from *Alice* and one from *Bob* to the feature vectors B_i^A or B_j^B. We denote the sets of all classification vectors as $\mathbf{C}^A$ (from $\mathbf{D}^A$) and $\mathbf{C}^B$ (from $\mathbf{D}^B$), respectively.

The two DOs then send their sets $\mathbf{C}^A$ and $\mathbf{C}^B$ of classification vectors together with encrypted record identifiers to the linkage unit (LU) to allow comparison of these vectors.

6. Using a privacy-preserving blocking protocol [1,9,14,18], record pairs that are similar according to their binary classification vectors C_i^A and C_j^B are inserted into the same block and compared by calculating the Hamming distance hd between them, where hd is defined as the number of bit positions where the value of $C_i^A[b] \neq C_j^B[b]$: $hd(C_i^A, C_j^B) = \sum_{b=1}^{n} |C_i^A[b] - C_j^B[b]|$.
7. Pairs of records that have a Hamming distance below a certain threshold value t, or equivalently a Hamming similarity value above a certain threshold, are considered to be matches and are added to the match set $\mathbf{M}$.

We next discuss key aspects of our protocol in more detail, and provide an analysis of the protocol with regard to privacy, complexity, and linkage quality.

4 Discussion and Analysis of the Protocol

As with any PPRL protocol, the overall usefulness of our protocol is determined by the linkage quality it achieves, the privacy protection it provides, and its computational complexity and parallelizability. The key component which determines all three aspects is the choice of parameter used for the SVMs employed in our protocol, and how these SVMs are trained, as we discuss in detail next.

4.1 Privacy Analysis

As with many other PPRL approaches, we assume all parties follow the *honest-but-curious* adversary model [22] in that they follow the steps of the protocol and do not behave in a malicious manner. We also assume the LU does not collude with either of the two database owners. However, each party aims to learn as much as possible about the other parties' sensitive data from the information it receives from the other parties. In our protocol, information is exchanged when the two DOs agree upon the set of attributes, A, to use for the linkage, which does reveal to each DO that they have this common set of attributes, but reveals nothing about sensitive values in these attribute. Exchanging the value of q, and the mapping of q-grams into binary feature vectors does not reveal any sensitive information.

In step 4 of the protocol the two DOs exchange the coefficients of their respective n trained linear SVMs. The main question here is if it would be possible for *Alice* to learn anything about the q-gram sets (and thus sensitive attribute values) used by *Bob* in his n training sets $\mathbf{T}_k^B$, and vice versa. As described in step 3 (a) of our protocol above, each DO keeps the random selection process secret, and therefore the other DO knows neither how many records were used for training each SVM (i.e. m is not known), how they were selected, the randomly set 0/1 class labels, nor any characteristics of these records. To learn anything about the binary feature vectors used for training a single SVM, an attacker would need to be able to reconstruct m (with m unknown) binary vectors

from the coefficients of this SVM, and repeat this process for all n SVMs. Because (ideally) sampling without replacement was used to generate the training sets, $\mathbf{T}_k^A$ and $\mathbf{T}_k^B$ respectively, there are no frequency distributions that could be exploited. Even the frequency distributions of coefficients in the trained linear SVMs does not reveal any direct information about the binary feature vectors used in training these SVMs.

These random choices done individually by the two DOs also reduce the risk of their sensitive data being revealed to the other DO in case of collusion between the linkage unit (LU) and one of the two DOs, as not knowing the random selection process does not allow a re-identification of sensitive attribute values. That said, if the LU knew the coefficients of the SVMs, then it could manufacture records to see whether they match the output of the SVMs.

At the end of step 5, both DOs send the binary classification vectors for their full databases to the LU. As with crypt-analysis attacks on Bloom filters [11,12,17], the LU could try to analyze the frequency distribution of 1-bits in these binary vectors, as well as the distribution of 1-bits for each position in these vectors. If we assume the set A contains several attributes, it is unlikely that more than one database record results in the same bit pattern (this would only happen if two database records have exactly the same values in all attributes in A). This is in line with record-level Bloom filters which have shown to be more secure than attribute-level Bloom filters [6,11,17].

Even if a frequency analysis would provide some information for the LU to investigate, unlike with Bloom filters based PPRL, where individual bit positions in a Bloom filter correspond to one or more hash-mapped q-grams, in our approach each bit position does not directly correspond to any q-gram but rather to the classification of two SVMs (one from each DO, as explained in step 5 of our protocol), i.e. the randomly trained SVMs are an intermediate step between database records and the binary classification vectors sent to the LU. Therefore, crypt-analysis attacks such as those successful on Bloom filters [11,12,17] would not be possible with our approach.

In step 5, the LU can learn the number of SVMs, n, used by each DO from the length of the classification vectors $\mathbf{C}^A$ and $\mathbf{C}^B$, but this will not reveal any sensitive information. To make our approach possibly more secure, in step 3 of our protocol, instead of taking the binary feature vectors to train the SVMs, the DOs could first encode feature vectors into Bloom filters (again resulting in binary vectors), and these Bloom filters could then be used as training records for the SVMs. However, given Bloom filters lead to false positives [16], this will likely result in a reduction in linkage quality.

Intuitively, assuming a set A of several attributes is used for the linkage, our proposed protocol exhibits improved privacy compared to Bloom filter based PPRL protocols because the random selection of the training data and the random selection of class labels provides an extra level of indirection between the sensitive attribute values and any information that is communicated between parties. We will experimentally evaluate the privacy of our approach in Sect. 5.

Table 1. Characteristics of data sets used in experiments.

Data set names	Number of records	Number of true matches
NCVR	224,073 / 224,061	124,597
DBLP-GS	2,616 / 64,263	get
ACM-DBLP	2,294 / 2,616	get

4.2 Computational Complexity

Assume $N = (|\mathbf{D}_A| + |\mathbf{D}_B|)/2$ is the average number of records in each of the two databases $\mathbf{D}^A$ and $\mathbf{D}^B$, and $G = (\sum_{\mathbf{Q}^A} |Q_i^A| + \sum_{\mathbf{Q}^B} |Q_j^B|/2N$ is the average number of q-grams generated for each record in step 1 of our protocol.

The communication complexity of exchanging parameter values between the two DOs is $O(1)$, while in step 4 each DO sends the coefficients of n trained SVMs to the other DO, leading to one message of size $O(nm)$ assuming the number of coefficients is of the order of training records m In step 5 of our protocol, each DO sends its set of N binary classification vectors to the LU, where each of those is of length n, leading to a communication complexity of $O(nN)$.

Peter: Alexis could perhaps help us here. The computational complexity of steps 1 and 2 of our protocol is both $O(GN)$, while training the n linear SVMs in step 3 has a complexity of $O(nm^2)$ assuming a quadratic complexity of training a SVM in the number of training records, m. In step 5, the generation of the N binary classification vectors (each of length n) is of $O(Nnm)$ if we assume each SVM has m coefficients and there are n SVMs. Finally, if the LU in step 6 applies a blocking approach which splits the databases into B equal sized blocks, then B blocks of size N/B need to be compared, resulting in $B * (N/B)^2$ calculations of Hamming distances on binary vectors of length n, leading to a complexity of $O(nN^2/B)$. Note that this comparison-complexity is the same as the comparisons of Bloom filter based PPRL approaches, however the length of the compared binary vectors (n) is likely to be less than the length of the Bloom filter vectors, leading to a faster comparison by the LU.

4.3 Linkage Quality

The quality of linkage achieved by our PPRL protocol depends mainly on how well the binary classification vectors created in step 5 represent the actual attribute values in their corresponding records. As the linear SVMs trained in step 4 are used to create these vectors, the quality of these SVMs on how well they classify ...

determined mainly by m and n, as well is choice of SVM kernel and other SVM parameters - Joly suggested $m = 32$?

sample size and num samples are main settings We experimentally ..

5 Experiments and Discussion

We used several real-world data sets, as summarized in Table 1, to evaluate our novel approach for PPRL and compare it with Bloom filter based PPRL.

The North Carolina Voter Registration 'NVCR' data sets[1] contain real voter records including names and addresses. We used two sub-sets of NCVR collected at different points in time, and extracted records about individuals that included changes between the two sub-sets (such as changed surnames or addresses, or corrected name variations and misspellings). The 'DBLP-GS' (DBLP linked with Google Scholar) and 'ACM-DBLP' (ACM Digital Library linked with DBLP) data set pairs both contain bibliographic records (details of publications and authors) and have previously been used in various record linkage studies [10].

As baseline, we used a state-of-the-art Bloom filter encoding approach [6,17,21] and following these earlier works set the Bloom filter length to $l = 1,000$ and the number of hash functions for Bloom filters such that their percentage of 1-bits was around 50%. For both Bloom filters and random projection hashing we converted attribute values into bigrams, i.e. $q = 2$. We employed a locality sensitive hashing (LSH) based indexing approach [9] to efficiently reduce the number the comparisons between Bloom filters to only those that potentially have a high similarity (i.e. pairs of Bloom filters that do not have many 1-bit positions in common are not compared).

We implemented our approach as well as Bloom filter encoding using Python 2.7.3, and ran all experiments on a server with 64-bit Intel Xeon 2.4 GHz CPUs, 128 GBytes of memory and running Ubuntu 14.04. We used the Scikit-learn machine learning package [13] for the SVM classifier. To facilitate repeatability, the programs and test data sets are available from the authors.

Quality Evaluation how we set parameters BF len, um hash functions num svm, sample size, lin kernel, C=0.01, 0.1,1

how to measure linkage quality: prec, recall, f-measure

plots using NCVR balanced, as well as small bibliographic data sets 3 BF plots for different BF len values (500, 1000, 2000), several RP bars for different number of SVM/sample size values (for small data sets include both with / without replacement sampling)

Scalability Evaluation bar plots for NVR different sizes (10K, 33K, 100K, 333K, 1M), where x-axis is data set size, y-axis is F-measure (one plot) or stacked bar plots with encoding / blocking and comparison times (2nd plot) each of these bar plots has one bar for BF and one for best RPH (same parameters across data sets)

Privacy Evaluation While a variety of methods have been proposed to measure privacy for PPRL (such as information gain, disclosure risk and probability of suspicion) [21], no single privacy measure for PPRL is currently available.

Our aim here is to compare the privacy provided by our novel approach compared to Bloom filter based PPRL approaches. These are known to have

[1] `http://dl.ncsbe.gov/`

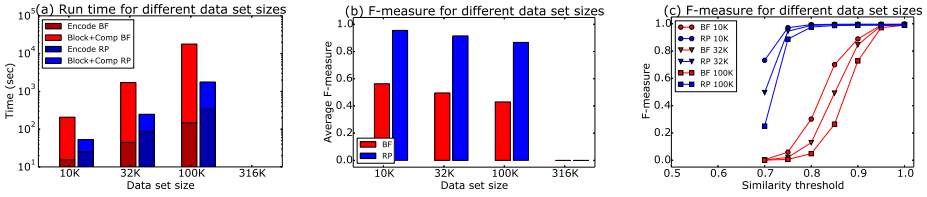

Fig. 2. Scalability results on sub-sets of the 'NCVR' data set with different numbers of records and well performing encoding parameter settings for the two encoding techniques (BF of length 1,000 and RP with 50 SVMs and sample size of 100. Our proposed approach outperforms BF both with regard to run time as well as linkage accuracy. For run-time, as can be seen in plot(a), RP is over one magnitude fast than BF, and most of the time is spent in the encoding phase while the blocking and comparison phase is faster due to the shorter bit vectors used by RP. And as plots (b) and (c) show, RP also achieves significantly higher linkage quality over a wider range of similarity thresholds (these thresholds set if a compared pair of bit vectors is classified as a match or a non-match).

vulnerabilities for certain parameter settings as has been shown in several recent crypt-analysis attacks [11,12,17]. These attacks are based on the frequency distributions of bits in Bloom filters. A certain bit position in a set of Bloom filters that contains a 1-bit for many Bloom filters very likely corresponds to a q-gram that occurs often in a certain attribute in a population database. This information can be used to iteratively map individual bit positions to certain q-grams [11,12], thereby revealing details of the encoded attribute values.

Therefore, the distribution of 1-bit frequencies in a set of binary vectors (Bloom filters or the classification vectors C_i^A and C_j^B in our approach) should be as close to a set of uniformly distributed random vectors (with a certain percentage of 1-bits).

We analyzed these 1-bit frequencies experimentally and present selected results in Fig. 3. We used 100,000 records from the NCVR data sets for these experiments, set the Bloom filter length and number of SVMs to 1,000 to allow comparison, the SVM sample size $m = 50$ and the number of hash functions for Bloom filters such that the percentage of 1-bits was around 50%. As baseline we also generated random binary vectors with 50% 1-bits uniformly distributed. To evaluate the correlation between the number and frequency of attribute values and corresponding 1-bit patterns, we only used the one or five most frequent value(s) of one attribute (first name).

As can be seen from Fig. 3, our random projection hashing approach leads to 1-bit frequency distributions much closer to uniform random, thereby making any frequency-based crypt-analysis attack more difficult compared to Bloom filter encoded values. The distributions for Bloom filters also show a set of bit positions where all records have a 1-bit, indicating to an attacker certain q-grams occur in all records (for the case where we only used the most frequent first name value). On the other hand, no such pattern is observable for random

projection hashing. Additionally, not visible in these plots, is that for Bloom filters commonly there were several bit positions with not only the same number of 1-bits, but also the same bit pattern over the set of encoded values, providing an attacker with information about correlated bit positions and therefore correlated q-grams. For random projection hashing, on the other hand, for all our experiments no bit pattern occurred more than once, and having a certain number of 1-bits occurring more than once was very rare.

Based on this frequency analysis, we conclude that crypt-analysis attacks that have shown to be possible on Bloom filter encoded data sets will be much more difficult on data sets encoded using our random projection hashing approach. We will investigate more formal methods for measuring and comparing the privacy of our approach as future work.

6 Conclusions and Future Work

We presented...

Peter: I don't think the other kernels will affect the privacy but they might affect the quality. Alexis might have insight into that. As future work we plan to explore the use of other SVM kernel functions and how these affect the linkage quality as well as the privacy of our approach, and to investigate other methods to more formally assess the privacy provided by our approach.

Peter: I'm not sure about this because then the LU could learn everything by building fake records an testing them. We aim to develop a multi-party version of our approach, where one central organization, such a government linkage unit (LU), is training a large enough number of SVMs on a large population database, and sends these SVMs to all database owners. The database owners can then use them to generate their binary classification vectors and send these back to the LU for calculating the similarities between records from different database owners. Such an approach does not require any communication between the database making the approach more secure.

References

1. Al-Lawati, A., Lee, D., McDaniel, P.: Blocking-aware private record linkage. In: IQIS. pp. 59–68. Baltimore (2005)
2. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
3. Boyd, J., Randall, S., Ferrante, A.: Application of privacy-preserving techniques in operational record linkage centres. In: Medical Data Privacy Handbook (2015)
4. Christen, P.: Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer (2012)
5. Clark, D.E.: Practical introduction to record linkage for injury research. Injury Prevention 10, 186–191 (2004)
6. Durham, E.A., Kantarcioglu, M., Xue, Y., Toth, C., Kuzu, M., Malin, B.: Composite bloom filters for secure record linkage. IEEE TKDE 26(12), 2956–2968 (2014)

7. Hall, R., Fienberg, S.: Privacy-preserving record linkage. In: Privacy in Statistical Databases, Springer LNCS 6344. pp. 269–283. Corfu, Greece (2010)
8. Joly, A., Buisson, O.: Random maximum margin hashing. In: IEEE CVPR. pp. 873–880. Colorado Springs (2011)
9. Karapiperis, D., Verykios, V.S.: A fast and efficient hamming lsh-based scheme for accurate linkage. Springer KAIS 49(3), 1–24 (2016)
10. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data and Knowledge Engineering 69(2), 197–210 (2010)
11. Kuzu, M., Kantarcioglu, M., Durham, E., Malin, B.: A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. In: PET. pp. 226–245 (2011)
12. Niedermeyer, F., Steinmetzer, S., Kroll, M., Schnell, R.: Cryptanalysis of basic Bloom filters used for privacy preserving record linkage. JPC 6(2) (2014)
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.: Scikit-learn: Machine learning in Python. JMLR 12, 2825–2830 (2011)
14. Ranbaduge, T., Vatsalan, D., Christen, P., Verykios, V.: Hashing-based distributed multi-party blocking for privacy-preserving record linkage. In: PAKDD (2016)
15. Randall, S., Ferrante, A., Boyd, J., Bauer, J., Semmens, J.: Privacy-preserving record linkage on large real world datasets. JBI 50, 205–212 (2014)
16. Schnell, R., Bachteler, T., Reiher, J.: Privacy-preserving record linkage using Bloom filters. BMC Med Inform Decis Mak 9(1) (2009)
17. Schnell, R., Borgs, C.: Randomized response and balanced bloom filters for privacy preserving record linkage. In: ICDMW. Barcelona (2016)
18. Vatsalan, D., Christen, P.: Sorted nearest neighborhood clustering for efficient private blocking. In: PAKDD. pp. 341–352. Gold Coast, Australia (2013)
19. Vatsalan, D., Christen, P.: Scalable privacy-preserving record linkage for multiple databases. In: ACM CIKM. Shanghai (2014)
20. Vatsalan, D., Christen, P.: Privacy-preserving matching of similar patients. JBI 59, 285–298 (2016)
21. Vatsalan, D., Christen, P., O'Keefe, C.M., Verykios, V.S.: An evaluation framework for privacy-preserving record linkage. JPC 6(1) (2014)
22. Vatsalan, D., Christen, P., Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. Elesevier IS 38(6), 946–969 (2013)

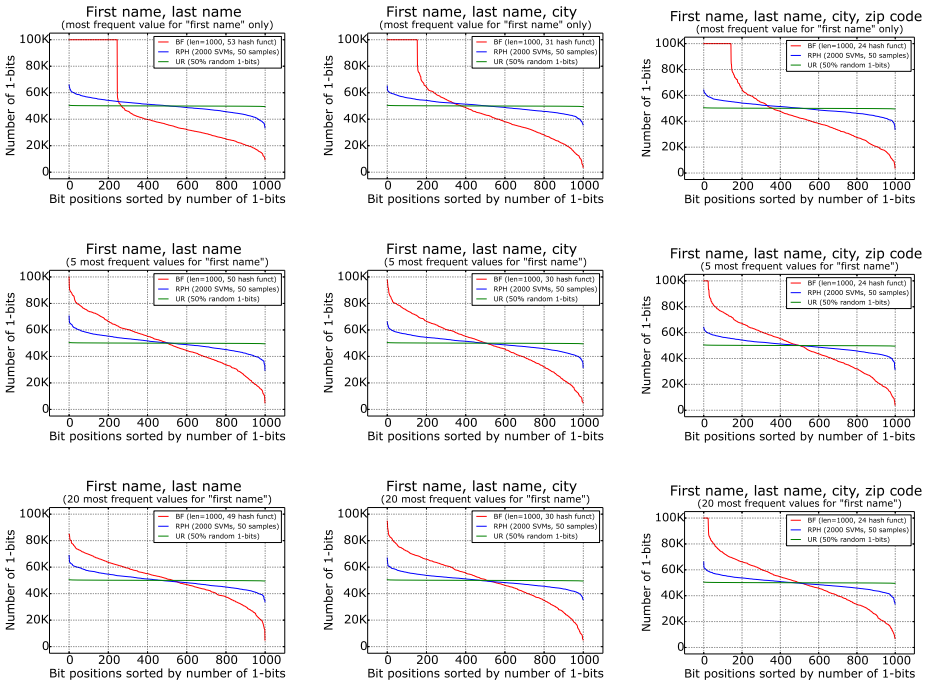

Fig. 3. *Comment Peter: I don't think we need to keep the plots for both 5 and 20, they are too similar, so best only to keep 20 I think* The distribution in descending order of the percentage of 1-bits at each position in each vector for randomly generated 1/0 vectors (the ideal baseline in green), random projection hashing (blue), and Bloom filters (red). A sub-set of 100,000 records from the NCVR data set was used for these experiments, where for the 'First name' attribute only the one (top row) or five (bottom row) most frequent values were used. Because the blue distribution is very similar to the green baseline, a frequency-based crypt-analysis attack would reveals little about the content of the values encoded in the corresponding binary vectors. In contrast, the more skewed distributions for Bloom filters could more likely permit the re-identification for certain q-grams, and thus attribute values, for certain bit positions.

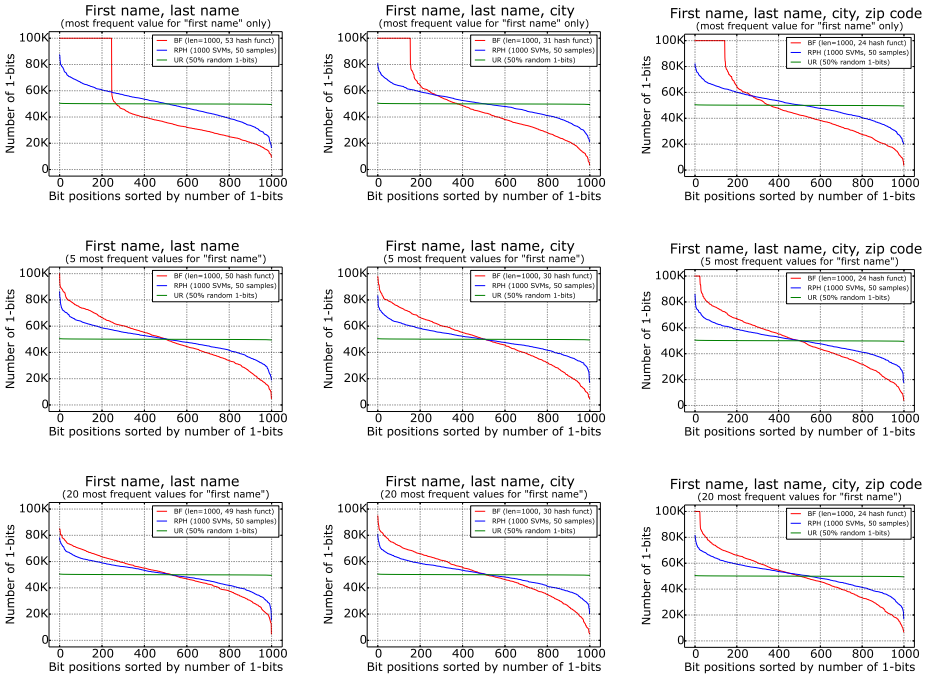

Fig. 4. Using concatenation of SVMs instead of XORing.