Efficient Privacy-Aware Record Integration

Mehmet Kuzu
Dept. of Computer Science
University of Texas at Dallas
Richardson, TX, 75080, USA
mehmet.kuzu@utdallas.edu

Murat Kantarcioglu
Dept. of Computer Science
University of Texas at Dallas
Richardson, TX, 75080, USA
muratk@utdallas.edu

Ali Inan
Dept. of Computer Engineering
Isik University
Istanbul, Turkey
ali.inan@isikun.edu.tr

Elisa Bertino
Dept. of Computer Science
Purdue University
W. Lafayette, IN, 47907, USA
bertino@cs.purdue.edu

Elizabeth Durham
Dept. of Biomedical Informatics
Vanderbilt University
Nashville, TN, 37232, USA
ea.durham@vanderbilt.edu

Bradley Malin
Dept. of Biomedical Informatics
Vanderbilt University
Nashville, TN, 37232, USA
b.malin@vanderbilt.edu

ABSTRACT

The integration of information dispersed among multiple repositories is a crucial step for accurate data analysis in various domains. In support of this goal, it is critical to devise procedures for identifying similar records across distinct data sources. At the same time, to adhere to privacy regulations and policies, such procedures should protect the confidentiality of the individuals to whom the information corresponds. Various private record linkage (PRL) protocols have been proposed to achieve this goal, involving secure multi-party computation (SMC) and similarity preserving data transformation techniques. SMC methods provide secure and accurate solutions to the PRL problem, but are prohibitively expensive in practice, mainly due to excessive computational requirements. Data transformation techniques offer more practical solutions, but incur the cost of information leakage and false matches.

In this paper, we introduce a novel model for practical PRL, which 1) affords controlled and limited information leakage, 2) avoids false matches resulting from data transformation. Initially, we partition the data sources into blocks to eliminate comparisons for records that are unlikely to match. Then, to identify matches, we apply an efficient SMC technique between the candidate record pairs. To enable efficiency and privacy, our model leaks a controlled amount of obfuscated data prior to the secure computations. Applied obfuscation relies on differential privacy which provides strong privacy guarantees against adversaries with arbitrary background knowledge. In addition, we illustrate the practical nature of our approach through an empirical analysis with data derived from public voter records.

EDBT/ICDT '13, March 18 - 22 2013, Genoa, Italy

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Security, integrity, and protection; H.3.3 [**Information Search and Retrieval**]: Information filtering; H.2.8 [**Database Applications**]: Statistical databases

General Terms

Security, Experimentation, Performance

Keywords

Privacy, Security, Record linkage, Differential privacy

1. INTRODUCTION

Modern decentralized environments make it easy for an individual's personal information to be dispersed among data sources managed by independent organizations. This phenomenon arises in wide array of societal contexts, and influences application domains ranging from homeland security to biomedical research [9]. To ensure accurate and robust analytics, it is critical to integrate information that corresponds to the same individual. Already, a substantial quantity of funding, policy, and research effort has been invested to develop record linkage techniques to support this task [9]. At the same time, such techniques have traditionally been applied on personal identifiers (e.g., forename), which can lead to concerns about personal privacy.

To mitigate such concerns, the past decade has witnessed a rise in private record linkage (PRL) frameworks. These techniques strive to enable data integration without revealing the explicit identifiers of the corresponding individuals. To achieve their goal, PRL protocols must be resistant to typographical errors (e.g., "john" vs. "jonh") that can arise in real world data sources [14, 15]. Previous PRL algorithms [1, 16, 17, 25, 28] achieve this feat by computing the similarity of record pairs via two fundamental mechanisms: 1) secure multi-party computation (SMC) [2, 10] and 2) similarity preserving data transformation [24, 25, 30].

SMC approaches enable the identification of final match results without leaking information through accurate and secure similarity computations, but their current instantiations are computationally too expensive. According to a

recent survey [6], secure edit distance computations [2] require over two years on a commodity server to compare two datasets of 1000 strings each. Alternatively, approaches based on data transformation are more practical but they have several drawbacks. First, the transformed records can leak uncontrolled information to an adversary that, under some circumstances, can be leveraged to attack the system and infer the original sensitive information [21]. Second, almost all transformation techniques map records into a metric space through an approximation strategy. This process induces some information loss, which in turn introduces false positives (i.e., a pair of records that are unmatched in the original space, but is classified as a match in the transformed space). For instance, in [24] records are embedded into Euclidean space by reporting their distance to a set of reference points. The approach preserves similarity to a certain extent, but strings that are expected to be highly dissimilar may become sufficiently close to look like they match [24].

The primary goal of this paper is to enable accurate and efficient similarity computations for practical PRL with strong privacy guarantees on individual records. To achieve this goal, we propose a three-step process. First, we construct local "blocks" on the data sources to eliminate secure similarity computations for record pairs that are expected to be non-matches. To form blocks on distinct data sources without any information leakage, we partition the data space using publicly available identifiers[1]. After blocking, local blocks are released in such a way that their release leaks only differentially private information. Finally, we apply an efficient SMC technique on the reduced set of candidate pairs to identify which records are sufficiently similar to match. In this setting, any information that is useful to the adversary is controlled according to the rigorous definition of differential privacy (DP) [7] and secure similarity computations are performed on the original data space. In summary, there are several notable contributions of this study:

Approximate Matching with Controlled Leakage: Almost all practical PRL approaches that enable approximate record matching leak information (e.g., relative similarity between each record) without any precise control. Although few studies [16, 17] controls this leakage, they are designed for numeric data only. Hence, they cannot handle approximate matching for string fields (e.g., forename) that are common in real data sources. In this study, we propose a practical PRL protocol that enables approximate matching for both string and numeric fields with controlled and limited information leakage. Specifically, any information leaked prior to secure similarity computations is obfuscated according to DP to ensure strong privacy guarantees.

Space Partitioning using Public Resources: Record linkage is typically performed with personal identifiers (e.g., names and addresses), that can be found in publicly available resources such as voter registration lists. We propose utilization of such resources to improve the linkage efficiency without leaking any information. Specifically, we divide the data space using public identifiers to form identical space representation for distinct parties. This enables the construction of common blocks on separate data sources to reduce the amount of secure similarity computations.

[1]If public identifiers are unavailable, both data sources are mapped to a single block. This increments the computational burden but the proposed approach is still applicable.

Efficient SMC Protocol for PRL: We propose an efficient SMC approach based on a semantically secure Paillier encryption scheme [22] for secure similarity computations.

The remainder of the paper is organized as follows. Section 2 formulates the problem and Section 3 provides background information. Then, we present our solution in Section 4 and its information leakage in Section 5. In Section 6, we report our experimental analysis. Finally, we review related work in Section 7 and conclude in Section 8.

2. PROBLEM FORMULATION

Imagine Alice and Bob want to identify and share similar records in their respective datasets A and B without revealing the identities of the corresponding individuals. In this context, the identification of similar records is equivalent to building a classifier that labels record pairs as "Match" or "Non-match". In the private record linkage problem, an accurate classifier is assumed to be available. Hence, the goal is to execute the classifier in a private, accurate and efficient manner. Without loss of generality, we assume that A and B have the same schema with attributes $\{R_1, ..., R_d\}$. We also assume that the characteristics of the classifier (f) can be expressed as a decision rule.

Decision Rule (f) : Let $sim: dom(R_i) \times dom(R_i) \mapsto \mathcal{R}$ be a similarity function defined over the domain of R_i; α_i be the weight for attribute R_i and θ be the matching threshold such that $\theta > 0$. Then, $f: a(R_1, ..., R_d) \times b(R_1, ..., R_d) \mapsto$ {match, nonmatch} is defined as follows :

$$f(a,b) = \left\{ \begin{array}{cc} match, & \text{if } \sum_{i=1}^{d} \alpha_i \cdot sim(a.R_i, b.R_i) \geq \theta \\ nonmatch, & \text{otherwise} \end{array} \right\}$$

The above definition of f is based on the Fellegi-Sunter (FS) model [11] which is a commonly applied decision rule construction mechanism for record linkage [9]. In the FS model, the attribute weight of R_i (α_i) is identified using the conditional probabilities of agreement on R_i given the match status. Suppose that m_{R_i} and n_{R_i} represent the conditional probabilities, such that $m_{R_i} = P[a.R_i = b.R_i \mid a \in A,\ b \in B,\ (a,b) \in Match]$ and $n_{R_i} = P[a.R_i = b.R_i \mid a \in A,\ b \in B,\ (a,b) \in Nonmatch]$. Then, α_i is defined as follows:

$$\alpha_i = log(\frac{m_{R_i}}{n_{R_i}}) - log(\frac{1 - m_{R_i}}{1 - n_{R_i}})$$

Once f is executed in a privacy preserving manner, Alice and Bob share the matching record pairs with each other. Any record of A and B that does not satisfy the match condition remains undisclosed.

In the protocol setting, Alice does not have any access to dataset B and Bob does not have any access to dataset A. The computation of f is facilitated through a third party, Charlie[2], who helps the execution of the protocol without learning any record from either A or B. We assume the participants of the protocol are semi-honest. As such, Alice, Bob and Charlie will follow the protocol as it is defined, but they may try to infer private information based on the messages they receive during the execution. We further adopt the standard assumption that participants do not collude.

[2]The utilization of the third party is a common practice for privacy preserving data sharing [6].

3. BACKGROUND

In Section 3.1, we briefly outline some basic concepts related to differential privacy. Then, in Section 3.2, we summarize the properties of Paillier cryptosystem.

3.1 Differential Privacy

Every privacy protection mechanism is vulnerable to some type of background knowledge known to a hypothetical adversary [7]. As a result, undesirable disclosures for an individual can occur regardless of whether or not the corresponding individual is included in the attacked database. Thus, it has been recommended that, instead of tailoring privacy definitions against different types of background knowledge, a data owner should minimize the risk of disclosure that arises from participation in a database. This notion is captured by the differential privacy (DP) protection mechanism [7]. DP is designed to address the case of statistical databases where users are allowed to ask only aggregate queries.

To protect privacy, DP adds random noise to each query result. The magnitude of the noise depends on a privacy parameter ϵ and sensitivity of the query set $\mathcal{Q}$. Denoting the response to query Q over data set T with Q^T, sensitivity [8] is defined as follows:

L_1-sensitivity : Over any sibling datasets T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record, the *L_1-sensitivity* of query set, $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$, is measured as:

$$S(\mathcal{Q}) = \max_{\forall T_1, T_2} \sum_{i=1}^{q} |Q_i^{T_1} - Q_i^{T_2}|.$$

Theorem 3.1 provides a sufficient condition to satisfy ϵ-differential privacy [8].

THEOREM 3.1. *Let $\mathcal{Q}$ be a set of queries, and $S(\mathcal{Q})$ be the L_1-sensitivity of $\mathcal{Q}$. Then, ϵ-differential privacy can be achieved by adding random noise X to each query result (i.e., $Q_i^T \leftarrow Q_i^T + X$), where X is a random, i.i.d. variable drawn from a Laplace distribution with magnitude $b \geq S(\mathcal{Q})/\epsilon$.*

3.2 Paillier Cryptosystem

Paillier cryptosystem [22] is a semantically secure asymmetric encryption scheme with some homomorphic features. Homomorphic features enable execution of certain operations on the ciphertexts as if they are performed on their plain versions. In the Paillier cryptosystem, if the same message is encrypted multiple times, the ciphertexts are not equal with very high probability. More formally, let $Enc_{K_{pub}}$ and $Dec_{K_{priv}}$ be the Paillier encryption and decryption functions with keys K_{pub} and K_{priv}, m_1 and m_2 be messages, c_{m_1} and c_{m_2} be ciphertexts such that $c_{m_1} = Enc_{K_{pub}}(m_1)$, $c_{m_2} = Enc_{K_{pub}}(m_2)$. Then, $c_{m_1} \neq c_{m_2}$, even if $m_1 = m_2$ with very high probability. Furthermore, the Paillier cryptosystem is homomorphicly additive. Given ciphertexts c_{m_1}, c_{m_2} and a constant γ, there exists efficient algorithms to compute the encryption of $m_1 + m_2$ and $\gamma \cdot m_1$. We represent homomorphic addition and constant multiplication by operators $\oplus_h$ and $\odot_h$ respectively such that $Enc_{K_{pub}}(m_1 + m_2) = c_{m_1} \oplus_h c_{m_2}$ and $Enc_{K_{pub}}(\gamma \cdot m_1) = \gamma \odot_h c_{m_1}$.

4. OVERVIEW OF THE SOLUTION

In this section, we provide an overview of our solution, which is visually summarized in Figure 1. It consists of three main steps: 1) local block construction, 2) block release and 3) integration via SMC. We present each step in detail through Sections 4.1 to 4.3. In the blocking step, a data partitioning is performed to reduce the number of candidate record pairs for linkage evaluation. Then, Alice release her blocks with encrypted contents to Bob. Finally, the decision rule for the record linkage is executed over the records of corresponding blocks of A and B through SMC.

4.1 Local Block Construction

The local block construction phase partitions the records into blocks to eliminate secure similarity computations for record pairs that are expected to be non-matches. To construct blocks on distinct data sources without leaking any private information, our model utilizes publicly available information such as voter registration lists. Specifically, the data space is partitioned using public identifiers (e.g., forenames) to generate an identical space representation for both Alice and Bob who forms blocks on their respective datasets using this representation[3]. In this phase, Charlie clusters public identifiers to form a global basis for block construction. Once Charlie shares this basis with Alice and Bob, they map their records to the blocks according to the basis. More formally, suppose $Id_{pub} = \{I_i, \ldots, I_j\}$ represents the set of publicly available identifiers that are included in the schema of the datasets to be linked and $\{\alpha_i, \ldots, \alpha_j\}$ represents their Fellegi-Sunter weights. Then, the identifier I_p with the maximum weight in Id_{pub} is selected as the partitioning identifier such that $I_p = \arg\max_{I_k \in Id_{pub}} \alpha_k$

Once I_p is selected, Charlie partitions the values that are in the public domain of I_p. Suppose $domain(I_p)$ represents the set of values in the public domain of I_p, $sim(a, b)$ represents the similarity between a and b where $a, b \in domain(I_p)$ and n_B represents the desired number of blocks. Then Charlie applies agglomerative hierarchical clustering [29] on $domain(I_p)$. Specifically, each value in $domain(I_p)$ is initially specified as a separate cluster. Then closest clusters are successively combined until the number of clusters is reduced to n_B. In this setting, closeness between clusters C_ρ and C_ℓ denoted as $closeness(C_\rho, C_\ell)$ is computed as follows:

$$closeness(C_\rho, C_\ell) = \frac{\sum_{a \in C_\rho} \sum_{b \in C_\ell} sim(a, b)}{|C_\rho| \cdot |C_\ell|}$$

Once values in $domain(I_p)$ are placed into clusters C_1 to C_{n_B}, Charlie transfers the populated clusters to the data owners as a global block basis. Then, data owners locally map their records to these clusters. Specifically, suppose r_i is a record in a local dataset with value $r_i.I_p$ on the partitioning identifier and C_ℓ is a cluster that contains public domain values $\{v_\ell[1], ..., v_\ell[z]\}$. Then the similarity between r_i and C_ℓ denoted as $sim_{cls}(r_i, C_\ell)$ is computed as follows:

$$sim_{cls}(r_i, C_\ell) = \max_{v_\ell[j] \in C_\ell} sim(r_i.I_p, v_\ell[j])$$

[3]Space partitioning with public identifiers is reasonable, because record linkage is typically performed with personal identifiers, which can be found in public resources such as voter registration lists and dictionaries.

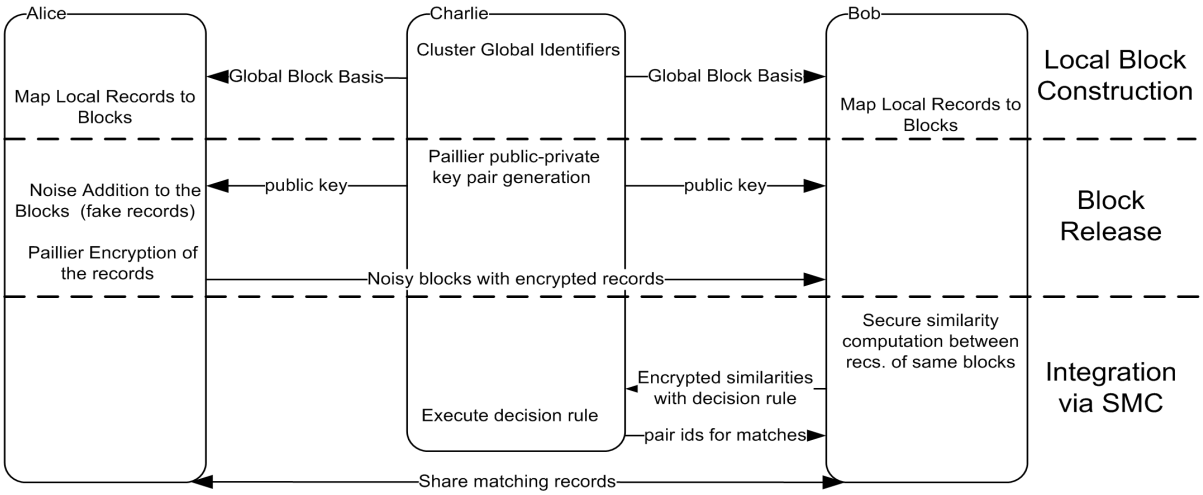

Figure 1: Outline of the proposed PRL protocol

After the similarities between r_i and clusters C_1 to C_{n_B} are computed, r_i is mapped to the most similar cluster. In this setting, Alice and Bob map each record in their respective datasets A and B to the available clusters. After mapping, each cluster C_ℓ with records of A (B) constitutes a local block for A (B).

We note that our model is applicable even when public identifiers are unavailable. In such a case, each distinct record in A and B is mapped to a single local block. Our experimental analysis (see Section 6) quantifies the additional cost of resource consumption such a practice entails.

4.2 Block Release

The local block construction results in blocks, such that the linkage decision rule will be evaluated only among records of the same blocks via SMC. To do so, blocks should be released by at least one data owner. Without loss of generality, suppose Alice releases her blocks to Bob after encrypting their contents. Although the records are in their encrypted form, direct release of the blocks discloses the number of records in each block. Hence, Bob may infer some information regarding the records of Alice by inspecting their distribution. In particular, Bob can compare their distribution against his data that are mapped to the same blocks.

To ensure individual privacy against any inference due to block release, our protocol protects blocks with a strong privacy protection mechanism called differential privacy (DP). Instead of releasing the original blocks of A, another version perturbed with DP is passed to Bob. Hence, Bob cannot infer the existence of a particular individual in A by the guarantees provided by DP. More formally, let C_i be a block label known to both Alice and Bob, and suppose A is divided into blocks as described in Section 4.1. Then, a differentially private block release corresponds to the issuing count queries of the following form:

select count(*) from A where Block = C_i

Given a set of queries $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$, differential privacy adds noise drawn from Laplace distribution with magnitude b to the true response value. b is determined by two parameters: 1) a privacy parameter ϵ and 2) the sensitivity of the query set $S(\mathcal{Q})$. We assume ϵ is set by the data owner (i.e., Alice). To determine $S(\mathcal{Q})$, we need to identify the sensitivity of block release. In this context, it is known that each query Q_i has a disjoint range by the construction of the blocks (e.g., each record is placed into a single block). Hence, with a single record update, we can change the result of at most two count queries by a magnitude of at most one.

THEOREM 4.1. *Let $\mathcal{Q}$ be the query set necessary for block release, then $S(\mathcal{Q}) = 2$.*

PROOF. Let r be an arbitrary record of dataset T in block C_i. Suppose some sibling dataset T' contains record r' instead of r, which is placed in block C_j. Then, there are two mutually exclusive cases.

Case I ($C_i = C_j$): The response to each query over T and T' will be the same (i.e., $\forall_{Q_z \in \mathcal{Q}} Q_z(T) = Q_z(T')$). Hence, $\sum_{z=1}^{|\mathcal{Q}|} |Q_z^T - Q_z^{T'}| = 0$.

Case II ($C_i \neq C_j$): Responses to queries Q_i and Q_j both will differ by 1 over T and T' (i.e., $|Q_z(T) - Q_z(T')| = 1$ for $z = i, j$). Hence, using $I(.)$ as an indicator function:

$$\sum_{z=1}^{|\mathcal{Q}|} |Q_z^T - Q_z^{T'}| = \sum_{z=1}^{|\mathcal{Q}|} 0 \times I(z \neq i \ \wedge \ z \neq j) + 1 \times I(z = i) + 1 \times I(z = j) = 2.$$

Therefore, $S(\mathcal{Q}) = \max[0, 2] = 2$ □

Since $S(\mathcal{Q}) = 2$, Alice adds Laplace noise to each partition with $b = 2/\epsilon$. It should be noted that Laplace noise can take on positive and negative values and is not necessarily integral.[4] Positive noise is incorporated by adding fake records to the dataset, while negative noise requires suppressing original records. For instance, in Figure 2-a, a_3 is removed from block C_1 because random noise is equal to -1. On the other hand, fake records r_1 and r_2 are added to C_2 and C_3, respectively, because random noise is equal to 1 for each. The generation of fake records and the encryption of block contents are discussed in Section 4.3.

[4]Integrality can be addressed by rounding-up to the closest integer value.

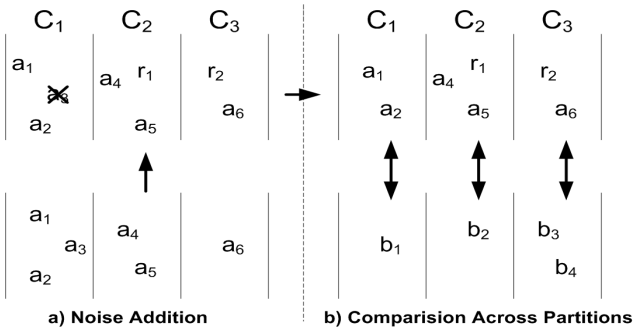

Figure 2: Block release

Once noisy blocks are transferred to Bob, linkage decision rule is evaluated securely (see Section 4.3) between record pairs of the same blocks. For illustration, in Figure 2-b, notice $b_1 \in B$ is compared with only $a_1, a_2 \in A$, whereas the suppressed record $a_3 \in A$ is not compared with any record in B. Similarly, $b_2 \in B$ will be compared against $a_4, a_5 \in A$ and fake record r_1.

In the record linkage setting, it is important to recognize that the effect of positive and negative noise is different. Consider, for each unit of positive noise, an additional fake record is inserted into a block, which causes more comparisons to be performed. On the other hand, negative noise results in suppressed records (e.g., a_3) which cannot appear in the final match set, even if there is a corresponding record in B. As a result, negative noise leads to degradation in the quality of the record linkage in terms of lower recall. Therefore, we assume Alice and Bob prefer will fake over suppressed records. Next, we show how this preference can be realized using a parameter, w_n, that indicates how many comparisons of [fake, original] record pairs the parties are willing to tolerate to prevent a single suppression.

Let $E[Y]$ represent the average number of records in a single block of dataset B. Bob needs to perform approximately $E[Y]$ comparisons for each fake record Alice inserts into her dataset. $E[Y]$ depends only on the total number of records in B and the number of blocks, neither of which differ between two sibling datasets. This implies zero L_1-sensitivity for $E[Y]$ release, such that Bob can release $E[Y]$ without adding noise. We use $E[Y]$ and w_n to determine the optimal noise addition strategy.

As explained in Section 3.1, ϵ-differential privacy is satisfied if the noise is drawn from a Laplace distribution with magnitude $b \geq S_{L_1}(\mathcal{Q})/\epsilon$. There is no restriction on how the mean μ is selected. Hence, in our solution, we select μ in order to (i) minimize the expected number of comparisons and (ii) optimize the number of suppressed records according to w_n[5]. If $w_n >> E[Y]$, the number of suppressed records will be minimized and the matching quality will be enhanced considerably at the expense of some additional [fake, original] record comparisons. Figure 3 provides a visual depiction of how the shifted mean will affect the noise distribution.

The cost associated with every unit of negative noise is proportional to w_n, while the cost for every unit of positive noise is proportional to $E[Y]$. Therefore, the optimal mean

[5]Suppression can be eliminated only if $\mu = \infty$. Therefore, we use w_n as a control parameter.

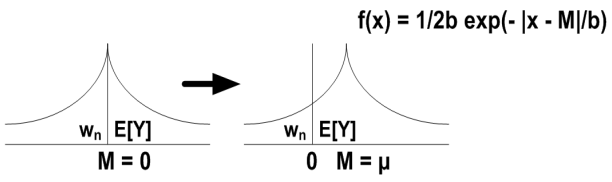

Figure 3: The mean shift for optimal cost

is the one that minimizes the following objective function:

$$\arg\min_{\mu} \frac{w_n}{2b} \int_{-\infty}^{0} |x| e^{-|x-\mu|/b}\, dx + \frac{E[Y]}{2b} \int_{0}^{+\infty} |x| e^{-|x-\mu|/b}\, dx$$

THEOREM 4.2. *Let $Lap(\mu, 2/\epsilon)$ represent the distribution of Laplace noise for ϵ-differential privacy. Then, the optimal mean (μ) for record linkage is*

$$\mu = -b \ln(\frac{2E[Y]}{E[Y] + w_n})$$

PROOF.

$$u = \tfrac{w_n}{2b} \int_{-\infty}^{0} |x| e^{-|x-\mu|/b}\, dx \; + \; \tfrac{E[Y]}{2b} \int_{0}^{+\infty} |x| e^{-|x-\mu|/b}\, dx$$

$$\tfrac{du}{d\mu} = \tfrac{d}{d\mu}[\tfrac{be^{-\mu/b}}{2}(w_n + E[Y]) + E[Y]\mu] = 0 \quad \wedge \quad \tfrac{d^2u}{d\mu} > 0$$

$$\implies \mu = -b \ln(\tfrac{2E[Y]}{E[Y]+w_n}) \quad \square$$

Once the cardinalities of noisy blocks are determined via $Lap(\mu, b)$, Alice transfers the perturbed blocks with encrypted records to Bob. Note that, though the DP model is applied to compute count queries, Alice outputs noisy sets of ciphertexts (e.g., encrypted version of each record). This is due to the fact that it is computationally infeasible for Bob to infer any information from the noisy sets except for their counts. This protection is based on the properties of the semantically secure encryption scheme we adopt for the protocol, further details of which are presented in Section 4.3.

4.3 Integration via SMC

The SMC step involves private evaluation of the decision rule on encrypted pairs of records in the same block. The evaluation requires calculating similarity over every attribute of the pair. As such, it is necessary to securely compute the similarity between $r_i.R_\iota$ and $r_j.R_\iota$ for each attribute R_ι, where r_i and r_j are records of a candidate pair.

We adopt different similarity functions for different types of attributes. For string-based attributes (e.g., forename), we use Jaccard similarity on their bigram bit vectors. This measure has been widely used for approximate string matching in the literature [19]. For numeric attributes (e.g., age), we use normalized Euclidean distance [17]. Based on these measures, Alice and Bob can evaluate the decision rule securely using generic SMC circuit evaluation techniques [12]. However, generic SMC techniques are computationally too expensive. In this study, we propose a relatively more efficient solution that utilizes the third party Charlie.

To compute similarity functions, we employ the Paillier cryptosystem [22] (see Section 3.2). In our setting, Charlie

generates a Paillier private/public key pair and transfers the public key to Alice and Bob. Alice encrypts her records and Bob performs some homomorphic operations with this key. Prior to encryption, Alice forms bit vectors for string attributes. Initially, she forms a bit vector of zeros that contains a single bit for each distinct bigram. Then, the string is divided into its bigrams and the bit locations that corresponds to its bigrams are set to one. Finally, each bit of the vectors and the number of ones in the vectors are encrypted as demonstrated in Figure 4.

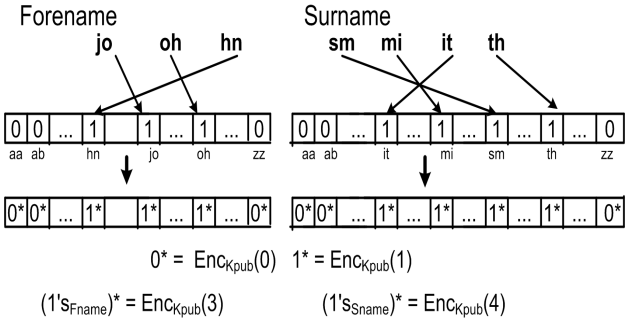

Figure 4: An example of the bitwise encryption

Formally, let R_s be a string attribute with corresponding bit vector representation $\vec{R_s}$ of size $|\vec{R_s}|$. Then the encrypted form of the attribute value for record r on R_s, denoted as $\pi_{r.R_s}$, consists of a bitwise encrypted vector ($\pi_{r.\vec{R_s}}$) and encrypted cardinality of the bit locations with value one ($\pi_{1's_{r.\vec{R_s}}}$) such that:

$$\pi_{r.R_s} = [\pi_{r.\vec{R_s}}, \pi_{1's_{r.\vec{R_s}}}]$$
$$\pi_{r.\vec{R_s}}[i] = Enc_{K_{pub}}(r.\vec{R_s}[i]) \quad \text{for} \quad 1 \leq i \leq |\vec{R_s}|$$
$$\pi_{1's_{r.\vec{R_s}}} = Enc_{K_{pub}}(\sum_{i=1}^{|\vec{R_s}|} r.\vec{R_s}[i])$$

For numeric attributes, Alice encrypts two pieces of information which are necessary for the computation of Euclidean distance. These are the original value itself and its square. Suppose R_n is a numeric attribute with the domain $[x, y]$ and $Enc_{K_{pub}}$ is the Paillier encryption function. Then the encryption of the attribute value of record r on R_n ($r.R_n$), and its square are formed as follows:

$$\pi_{r.R_n} = Enc_{K_{pub}}(r.R_n)$$
$$\pi_{(r.R_n)^2} = Enc_{K_{pub}}((r.R_n)^2)$$

After encryption of the original records, Alice adds the encrypted fake records to each block. To do so, Alice encodes the bit vector of a string attribute R_s of a fake record ω as a bit vector of all zeros (e.g., $\omega.\vec{R_s} = [0....0]$). Additionally, Alice encodes numeric attribute R_n of ω with range $[x, y]$ as $\omega.R_n = -(y-x)$. This encoding is repeated for all attributes of ω to prevent matching between an original and a fake record in the following steps of the protocol. Finally, Alice encrypts the encoded fake records.

To compute the Jaccard similarity between the bit vectors of strings $a_i.R_s$ and $b_j.R_s$, it is necessary to compute $c_\cap = |a_i.\vec{R_s} \cap b_j.\vec{R_s}|$ and $c_\cup = |a_i.\vec{R_s} \cup b_j.\vec{R_s}|$. These cardinalities can be computed on the encrypted data using the additive homomorphic property of the Paillier encryption scheme. Suppose $[\pi_{a_i.\vec{R_s}}, \pi_{1's_{a_i.\vec{R_s}}}]$ represents the encryption of attribute R_s for record a_i. Similarly, suppose $b_j.\vec{R_s}$ denotes the plain bit vector form of R_s for record b_j and $L_{b_j.\vec{R_s}} = \{\ell_1, .., \ell_n\}$ is the set of its bit locations with value one respectively. Then, Bob computes the following encrypted cardinalities for R_s:

$$\pi_{c_\cap} = \pi_{a_i.\vec{R_s}}[\ell_1] \oplus_h ... \oplus_h \pi_{a_i.\vec{R_s}}[\ell_n]$$

$$\pi_{(c_\cap + c_\cup)} = Enc_{K_{pub}}(\sum_{z=1}^{|\vec{R_s}|} b_j.\vec{R_s}[z]) \quad \oplus_h \quad \pi_{1's_{a_i.\vec{R_s}}}$$

$$\pi_{c_\cup} = \pi_{(c_\cap + c_\cup)} \oplus_h (-1 \odot_h \pi_{c_\cap})$$

For each numeric attribute R_n, Bob computes the encrypted form of the Euclidean distance between $a_i.R_n$ and $b_j.R_n$ ($[a_i.R_n - b_j.R_n]^2$) as follows:

$$\pi_{(a_i.R_n - b_j.R_n)^2} = \pi_{(a_i.R_n)^2} \oplus_h Enc_{K_{pub}}((b_j.R_n)^2) \oplus_h ((-2 \cdot b_j.R_n) \odot_h \pi_{a_i.R_n})$$

Once the encrypted cardinalities have been computed for each attribute of the candidate pair (a_i, b_j), Bob assigns a pair identifier p_{ij} to the pair. Then, he can send the cardinalities along with the pair identifier to Charlie. Since Charlie holds the private key to decrypt the encrypted cardinalities and the decision rule, he can identify matching pairs. To do so, Charlie can compute the Jaccard similarity for a string attribute R_s as follows:

$$sim_{R_s}(a_i.\vec{R_s}, b_j.\vec{R_s}) = \frac{Dec_{K_{priv}}(\pi_{(|a_i.\vec{R_s} \cap b_j.\vec{R_s}|)})}{Dec_{K_{priv}}(\pi_{(|a_i.\vec{R_s} \cup b_j.\vec{R_s}|)})}$$

For a numeric attribute R_n with range $[x, y]$, Charlie can compute similarity according to the normalized Euclidean distance as follows:

$$sim_{R_n} = 1 - Dec_{K_{priv}}(\pi_{(a_i.R_c - b_j.R_c)^2})/(y-x)^2$$

It is guaranteed that $sim_{R_s}(a_i.\vec{R_s}, b_j.\vec{R_s}) = 0$ for any string attribute R_s if a_i is a fake record. By construction, $a_i.\vec{R_s} = [0, \ldots, 0]$, and thus $a_i.\vec{R_s} \cap b_j.\vec{R_s} = 0$. Notably, by the encoding of numeric attributes of the fake records, similarity between any numeric attribute of a fake and a real record will always be non-positive.

Suppose that records a_i and b_j are compared using the SMC protocol presented above. Then, Charlie observes neither $a.R_\iota$ nor $b.R_\iota$ but $sim(a_i.R_\iota, b_j.R_\iota)$. To prevent leakage of any information except for the final match decision, we employ a random perturbation on the $sim(a_i.R_\iota, b_j.R_\iota)$ which we call *similarity blinding*. Employed perturbation is an SMC approach (e.g., [18, 27]) that relies on the impossibility of solving linear equations with too many unknowns.

Similarity Blinding: The linkage decision rule derived from the Fellegi-Sunter model [11] classifies a record pair (a_i, b_j) as a match if and only if $\sum_{z=1}^{d} \alpha_z \cdot sim(a_i.R_z, b_j.R_z) \geq \theta$. The basic idea behind similarity blinding is to perturb $sim(a_i.R_z, b_j.R_z)$ with random values so that Charlie

cannot infer its exact value. To do so without preventing correct evaluation of the decision rule, we must perturb the matching threshold, θ, accordingly. We perturb every pair of records (a_i, b_j) independently. Therefore, perturbed θ values will differ between distinct record pairs. The perturbed threshold of pair (a_i, b_j) is referred with θ_{a_i,b_j}.

Blinding is an incremental process performed by Bob over all attributes. θ_{a_i,b_j} is initially set equal to θ, such that manipulation of each attribute will further change its value. Once all attributes are blinded, Bob sends θ_{a_i,b_j} and the encrypted cardinalities to Charlie so that he can evaluate the decision rule correctly. We discuss the cases of a string attribute R_s and a numerical attribute R_n below.

Suppose R_s is a string attribute with weight α_{R_s} and $c_\cap$, $c_\cup$ denote $|a_i.\vec{R_s} \cap b_j.\vec{R_s}|$ and $|a_i.\vec{R_s} \cup b_j.\vec{R_s}|$ respectively, such that $sim(a_i.\vec{R_s}, b_j.\vec{R_s}) = c_\cap/c_\cup$. Note that, according to the protocol, Bob can compute the encryptions $\pi_{c_\cap}$ and $\pi_{c_\cup}$ of $c_\cap$ and $c_\cup$ respectively.

Now, to blind the similarity on R_s, Bob first generates random values r_{ij_1}, r_{ij_2} such that $r_{ij_2} \neq 0$ and $r_{ij_1}, r_{ij_2} \in Z_n$. Then, he updates $\pi_{c_\cap}$ and $\pi_{c_\cup}$ to ensure Charlie will observe the following blinded similarity value:

$$sim(a_i.\vec{R_s}, b_j.\vec{R_s}) + \frac{r_{ij_1}}{r_{ij_2}} = \frac{c_\cap}{c_\cup} + \frac{r_{ij_1}}{r_{ij_2}} = \frac{r_{ij_2} \cdot c_\cap + r_{ij_1} \cdot c_\cup}{r_{ij_2} \cdot c_\cup}$$

This is achieved through the following homomorphic operations:

$$\pi_{c_\cap} \leftarrow (r_{ij_2} \odot_h \pi_{c_\cap}) \oplus_h (r_{ij_1} \odot_h \pi_{c_\cup})$$
$$\pi_{c_\cup} \leftarrow r_{ij_2} \odot_h \pi_{c_\cup}$$

To ensure that the matching decision will not be affected, Bob also increments θ_{a_i,b_j} by $\alpha_{R_s} \cdot r_{ij_1}/r_{ij_2}$. Here, multiplication by α_{R_s} adjusts for the attribute weight of R_s.

The case of numeric attributes is simpler. Suppose R_n is a numeric attribute with weight α_{R_n} and range $[x, y]$. Bob generates a random number r_{ij} and homomorphicly adds it to the encryption of $(a_i.R_n - b_j.R_n)^2$:

$$\pi_{(a_i.R_n - b_j.R_n)^2} \leftarrow Enc_{K_{pub}}(r_{ij}) \oplus_h \pi_{(a_i.R_n - b_j.R_n)^2}$$

To ensure that the matching decision will not be affected, Bob also decrements θ_{a_i,b_j} by $\alpha_{R_n} \cdot (r_{ij}/(y-x)^2)$. In this equation, the multiplication by α_{R_n} adjusts for the attribute weight and the division by $(y-x)^2$ is for normalization.

The security provided by similarity blinding is based on a SMC approach that relies on the inability of solving a linear equation with more than one unknowns [18, 27]. In fact, attribute similarity extraction for a candidate pair from the perturbed similarities and perturbed threshold corresponds to solving a single linear equation with 'd' unknowns where 'd' represents the number of attributes. If $d = 1$, similarity blinding should be applied in a slightly different manner to generate at least two unknowns for a single equation. Specifically, suppose R_ι represents the single attribute, θ represents the matching threshold where $\theta > 0$, r_{ij_1} and r_{ij_2} are random numbers such that $r_{ij_1}, r_{ij_2} \in Z_n$. Then $sim(a_i.R_\iota, b_j.R_\iota)$ is set to $r_{ij_1} \cdot sim(a_i.R_\iota, b_j.R_\iota) + r_{ij_2}$ and matching threshold θ_{a_i,b_j} is set to $r_{ij_1} \cdot \theta + r_{ij_2}$ through the homomorphic operations. Note that, for each distinct candidate pair, a distinct matching threshold is generated with fresh randomness. Hence, each linear equation is independent from each other.

After Charlie executes the decision rule with perturbed similarities and matching thresholds, the pair identifiers of the matches are sent to Bob who maps them to the corresponding record identifiers. Finally, Alice and Bob share matching records with each other.

5. LEAKAGE ANALYSIS

We assume Alice, Bob and Charlie will follow the protocol honestly, but may try to infer private information based on messages they receive during the execution without collusion. The semi-honest model without collusion is a common assumption for multi-party protocols [21]. Next we summarize the information that our solution discloses to each of the participants and the inherent risks associated with such disclosures (if any).

Alice: This party does not receive any messages regarding Bob's dataset other than the final result.

Bob: This party observes the final results and the blocks of Alice's dataset with encrypted contents. It is computationally infeasible for Bob to infer anything from these records because they are encrypted with a semantically secure encryption scheme [13]. Hence, the only possible information leakage is the cardinality of A's blocks. Here, block cardinalities disclose the record distribution of A on the data space representation that is generated through clustering of publicly available identifiers (see Section 4.1). To ensure individual privacy against any adversary that may utilize this distribution, the proposed model perturbs block cardinalities according to the rigorous definition of differential privacy (DP). Hence, individual privacy is ensured by the strong guarantees of DP.

Charlie: This party participates in the blocking and the SMC steps. In the blocking step, he does not receive any messages regarding the datasets of Alice and Bob. In the SMC step, he does not receive any messages regarding the individual records in A or B but receives the perturbed similarities for each candidate pair. To keep the attribute similarities of candidate pairs secret from Charlie, the proposed model randomize both attribute similarities and the matching threshold in such a way that actual similarity extraction necessitates solving a single linear equation with at least two unknowns (see Section 4.3). Random perturbation that relies on the impossibility of solving linear equations with too many unknowns is a common SMC approach to hide the actual input [18, 27]. Here, Charlie executes the decision rule with perturbed input to identify if the received candidate is a match without learning actual similarities.

6. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation[6] of the proposed private record linkage protocol. In Section 6.1, we present the experimental design. Then, in Sections 6.2 and 6.3, we present the evaluation of blocking and SMC steps of the proposed scheme. Finally, we present the record linkage efficiency in Section 6.4.

[6]All experiments were conducted on a server with two quad-core 2.5 GHz Intel Xeon processor and 32 GB RAM.

6.1 Experimental Design

To perform the experimental analysis, we selected a publicly available dataset of real personal identifiers, derived from the North Carolina voter registration list (NCVR) [5]. We investigated the effectiveness of the scheme with forename (F), surname (S), city (C), race (R) and gender (G) attributes. To construct the record linkage decision rule with these attributes, we computed their Fellegi-Sunter (FS) weights (see Section 2): $\alpha_F = 0.3$, $\alpha_S = 0.35$, $\alpha_C = 0.22$, $\alpha_R = 0.06$ and $\alpha_G = 0.07$. Here, FS weights indicate the relative importance of the utilized attributes on the linkage decision.

For blocking evaluation, we initially formed a public dataset (P) of size 10,000 by random record selection from NCVR. Then, we generated 10 dataset pairs (A_i, B_i) to be linked such that $P \cap A_i = \emptyset$ and $P \cap B_i = \emptyset$. To form dataset pairs, we initially selected 10 datasets by randomly drawing 5,000 records for each from NCVR (excluding those in P), which we refer to as $A_1, \ldots, A_{10}$. For each A_i, we generated a partner B_i composed of 5000 records as well. Of these records, 4000 were randomly selected from NCVR (excluding those in A_i and P), while 1000 were randomly selected from A_i and subjected to random perturbation. For perturbation, we introduced a typographical error to each distinct attribute 25% of the time[7] via a publicly available typo generator [26] which produces a variety of spelling errors (e.g., insertions, deletions, and transpositions). The goal was to privately identify the 1000 matching records between A_i and B_i. We use reduction ratio (RR) and pair completeness (PC) as evaluation metrics for our blocking scheme. Specifically, suppose c is the number of candidate record pairs produced by the blocking scheme, c_m is the number of true matches among c candidate pairs, $n = |A_i| \cdot |B_i|$ is the number of all possible pairs and n_m is the number of true matches among all pairs. Then, RR and PC are defined as follows:

$$RR = 1 - c/n, \quad PC = c_m/n_m$$

For SMC evaluation, we investigated the computational requirement (i.e., time and transferred data) of the SMC protocol with respect to varying input size and number of attributes. Time is reported as the computation time for cryptographic operations[8] (e.g., encryption and homomorphic addition). Transferred data corresponds to the amount of data transmitted between two distinct parties during protocol execution. For the SMC experiments, we initially generated dataset pairs (A_i, B_i) from NCVR with different sizes and number of attributes. The default input size is $|A_i| = 5000$, $|B_i| = 5000$ and the default attributes include forename, surname, city, race and gender.

6.2 Blocking Evaluation

We analyzed the influence of the number of blocks (n_B), differential privacy parameter ϵ, suppression cost (w_n) for the weighted noise cost model and utilized public identifier (I_p) for data space partitioning on the blocking performance. The default parameters are as follows: $n_B = 500, \epsilon = 0.3$, $w_n = 5000$, $I_p = surname$. Below, we report the average measurements over the 10 dataset pairs.

Figure 5 summarizes the blocking performance for different numbers of blocks. As expected, when the number of blocks is small, RR is limited. As the number of blocks increases, RR increases with the cost of slight decrease in PC. In this set of experiments, 500 blocks provide approximately 98% savings in the number of comparisons with a cost of 4% loss in matching pair identification.

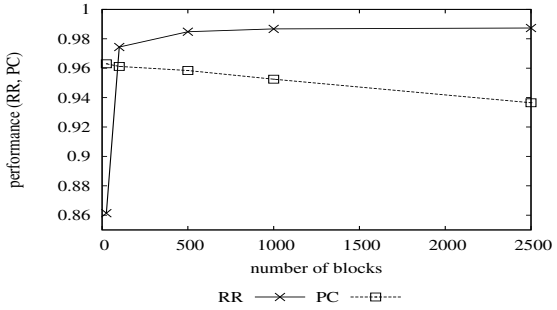

Figure 5: Effect of the number of blocks on the blocking performance

Figure 6 shows the effect of differential privacy parameter ϵ. PC is the same for all values of ϵ. This is because the default suppression cost leads to only positive noise and there is no suppression. On the other hand, RR improves with increasing ϵ. This is because larger ϵ results in less noise and less fake records in the blocks.

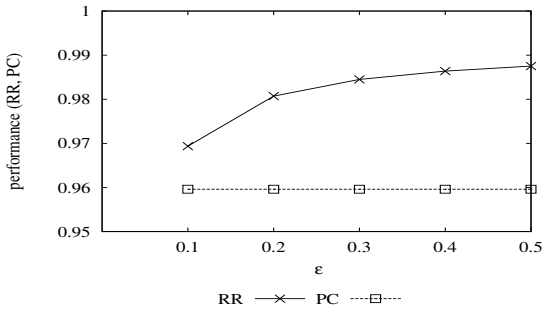

Figure 6: Effect of differential privacy parameter ϵ on the blocking performance

We proposed a weighted noise cost model for differential privacy because negative noise results in suppression of records and a subsequent decrease in matching quality. To mitigate this loss, we shifted the noise distribution according to a control parameter w_n (see Section 4.2). Figure 7 shows the effect of w_n on performance. For higher w_n, PC increases due to a reduced number of suppressions at the expense of some decrease in RR. After a certain point, additional increments in w_n lead to decreases in RR without any changes in PC. This is because we start adding only positive noise. In this setting, if standard Laplace mechanism with zero mean is applied, PC reduces approximately to 0.7. This considerably degrades the linkage quality and indicates the importance of the proposed weighted noise cost model.

[7]As pointed out in [20], statistics collected by Google for the search input "Britney Spears" indicates that users misspell the input 23% of the time. Hence, we chose 25% as a reasonable error rate for our experiments.

[8]For this set of experiments, Paillier encryption was performed with a 512-bit public key.

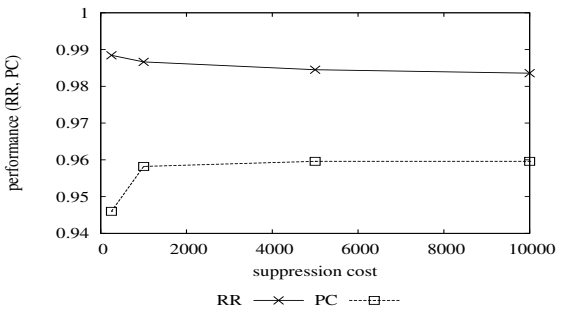

Figure 7: Effect of suppression cost for weighted Laplace noise on the blocking performance

Identifier	RR	PC
Forename	0.9822	0.9476
Surname	0.9845	0.9596
City	0.9740	0.9910

Table 1: Effect of selected public identifier on the blocking performance

Table 1 shows the effect of the selected public identifier (forename [F], surname [S], city [C]) for the global block basis construction. Here, RR depends on the attribute importance for the linkage decision. In fact, observed RR values are compliant with the FS weights of the attributes. Among the available identifiers, surname has the largest FS weight and provides the highest RR. In this setting, identifiers with very low FS weights (e.g., race) are not effective. The race provides approximately 40% RR while surname provides 98% RR. In this context, PC depends on the domain size of the attributes. The domain size for forename, surname and city in the public dataset are approximately 5000, 2500, and 600. With decreasing domain size, PC increases since global block basis represents space more precisely.

Blocking could be performed efficiently without significant computational cost. During blocking, Charlie forms the global block basis through a hierarchical clustering over the domain of the selected public identifier. The computational complexity of this operation is $O(m^2 \cdot log(m))$[9] where m represents the domain size of the selected identifier. Once global block clusters are formed, Alice and Bob map their respective records to these clusters. The computational complexity of map operation is $O(mn)$ where m and n correspond to the domain size of the selected public identifier in global and local datasets, respectively. In the default experimental setting, blocking is performed in a few minutes. The main bottleneck of the protocol is the SMC step, the performance of which is discussed in further detail below.

6.3 SMC Evaluation

Here, we present the computational requirements of our SMC protocol. In the proposed approach, Alice encrypts her records in each block and transfers the blocks to Bob. Then, Bob computes encrypted similarities for candidate pairs using the properties of the encryption scheme. Finally, Bob sends similarity values to Charlie who decrypts them to execute the decision rule. For this set of experiments, we applied blocking on the datasets with the default blocking setting. Then, we executed SMC on the blocks.

Figure 8 reports the execution time for encryption of the records in A_i and the amount of data transferred from Alice to Bob. With increasing input size, denoted as Λ ($\Lambda = |A_i| = |B_i|$ for this experiment), the computational cost increases almost linearly. This is because the number of encryptions performed, as well as the amount of data transferred, is proportional to $|A_i|$.

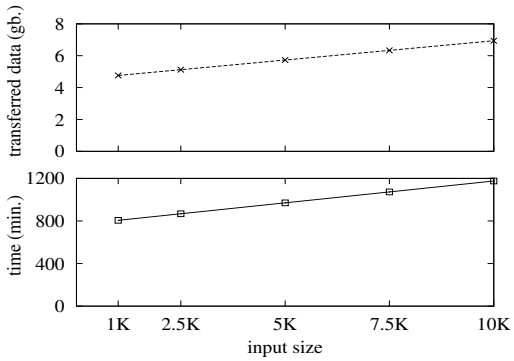

Figure 8: Load of Alice with distinct dataset size

Figure 9 reports the time required for Bob to perform the secure operations and the amount of data transferred from Bob to Charlie. As the number of records in A_i and B_i increases, computational demand increases proportionally to $|A_i| \cdot |B_i| \cdot RR$ where RR represents the reduction ratio. It is clear that blocking provides considerable savings in the computation. For instance, to link datasets of size ($10K \times 10K$), Bob transfers only 1.1 GBs of data instead of 55 GBs through a 98% RR achieved through the blocking step.

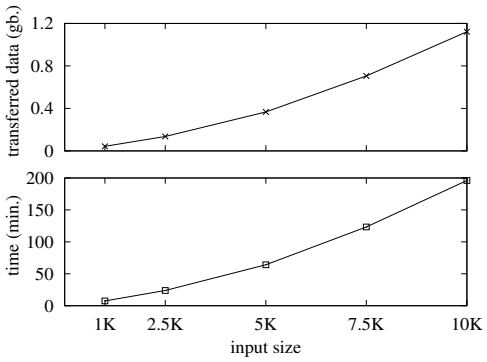

Figure 9: Load of Bob with distinct dataset size

Figure 10 shows the computational demand of Alice with distinct attributes (forename (F): string, surname (S): string, city (C): string, race (R): numeric, gender (G): numeric). As expected, the computational demand for numeric attributes is much less than for string attributes. This is because, for each string attribute, bitwise encryption is applied on their bit vector representation which corresponds to 676 (i.e., number of bigrams) distinct encryptions. For numeric attributes only a single encryption is performed.

[9] $O(m^2 \cdot log(m))$ complexity is due to the priority queue implementation of agglomerative hierarchical clustering.

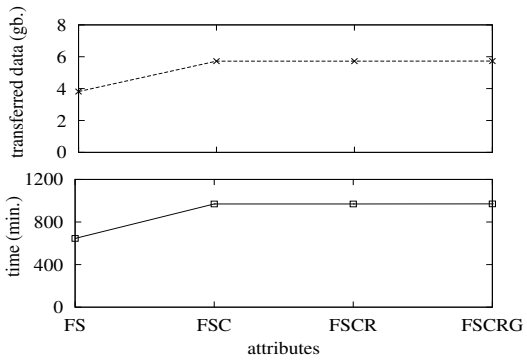

Figure 10: Load of Alice with distinct attribute set

Figure 11 demonstrates the load for Bob with distinct attributes. The amount of data transferred, as well as the amount of time spent on cryptographic operations increases almost linearly with the number of attributes. The string attributes do not dominate the computation for Bob. This is because Bob performs cryptographic operations for only a small number of bit locations, instead of all 676 distinct bit locations due to sparsity of the vectors (i.e., the bit vector of strings contains limited number of ones). In addition, Bob sends only encrypted cardinalities to Charlie instead of encrypted data. Specifically, he sends two cardinalities for each string attribute and one cardinality for each numeric attribute. Hence, the cost is similar for distinct data types.

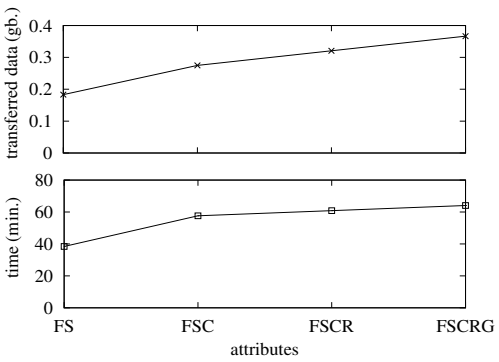

Figure 11: Load of Bob with distinct attribute set

Once Charlie receives the candidate pairs and the encrypted cardinalities for distinct attributes, he decrypts them for decision rule evaluation. Decryption of a single cardinality can be performed quite efficiently, on the order of roughly 4 milliseconds. Since this is a marginal cost in the overall protocol, we neglect Charlie's contribution to the computational complexity in the following assessment.

Finally, we present the performance of the approach for a scenario where publicly available identifiers are not available for blocking. In such a case, Alice and Bob map their records to a single block. The influence of blocking on the computational load of Alice and Bob in default experimental setting ($|A_i| = 5000$, $|B_i| = 5000$, $\epsilon = 0.3$, $n_B = 500$, $w_n = 5000$, $I_p = surname$) is presented in Table 2. Notice that when a single block is formed instead of 500 blocks, the computational load for Alice slightly decreases. This

	Default Blocks	Single Block
Time (A)	970 min.	205 min.
Storage (A)	5.7 G.	1.2 G.
Time (B)	123 min.	4166 min.
Storage (B)	0.7 G.	23.8 G.

Table 2: Influence of Blocking on the SMC Load

is because, the computational load of Alice is due to the record encryption, which depends only on the total number of records in A (including fake records). The noise is added to each block independently, so the number of fake records is linearly proportional to the number of blocks. On the other hand, the single block setting significantly increases the computational load for Bob. This is because the load is based on the amount of secure similarity computations between the record pairs which is proportional to $(1 - RR) \cdot |A_i| \cdot |B_i|$. With single block, RR is zero and the load of Bob is much higher in comparison to the setting with multiple blocks.

6.4 Record Linkage Efficiency

In the context of this study, we assume that a record linkage (RL) decision rule is available. Our goal is to provide a method for the private execution of the rule on the original data space in an efficient manner. For comparison, it is important to recognize that the most private and accurate techniques are based on SMC [6]. However, these solutions [2, 10] are computationally inefficient for large datasets. In a recent study [6], it was shown that a highly-cited SMC approach for PRL requires over 2 years to compare two datasets of 1K strings each. On the other hand, our approach can handle the same process in only several hours.

Computationally efficient PRL solutions [1, 24, 25, 30] generally execute the decision rule after applying a transformation to the data. Among such approaches, PRL techniques of [24] and [30] are the only ones that enable approximate data matching for records with both string and numerical attributes (see Section 7 for the details of other approaches). Both techniques execute the decision rule after transforming data into an Euclidean space through a reference value embedding technique, called SparseMap.

We investigated the linkage accuracy with our approach and SparseMap based approaches. To measure the accuracy, we utilized precision (i.e., the ratio of correctly linked records among all linkages) and recall (i.e., the ratio of correctly linked records to all matches between linked datasets) metrics. We investigated the success with forename and surname attributes as in [24]. To do so, we generated 10 dataset pairs with two attributes via the data generation mechanism of Section 6.1 such that each dataset contains 5000 distinct records. Then, we embedded records of each dataset into Euclidean space via SparseMap algorithm. With respect to the parameters of the algorithm, we followed the recommendations in [24]. We generated 30 reference sets to embed records into 30-dimensional space which provided the best accuracy in [24]. The length of the reference strings in the sets was 15, which was approximately equal to the average length of the strings (forename + surname) in the dataset to be embedded. With respect to the heuristics of the algorithm, we disabled both greedy resampling and distance approximation heuristics to improve the quality of the em-

beddings (see [24] for details). Once the records were embedded, we linked each record in A_i to the closest record in B_i using Euclidean distance in the new space. Then, we applied our algorithm on the original space. After blocking with default blocking parameters ($n_B = 500, \epsilon = 0.3$, $w_n = 5000$, $I_p = surname$), we computed Jaccard similarity between the candidate records pairs. According to the computed similarities, we linked each record in A_i to the closest candidate record in B_i.

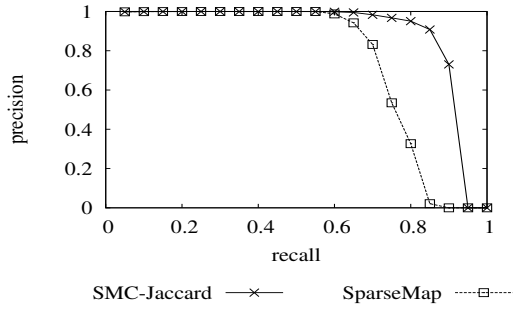

Figure 12: Record matching accuracy of the proposed approach in compare to SparseMap

In Figure 12, we report the accuracy of the linkage results of our approach and the SparseMap approach [24, 30]. The reported values are the averages over the 10 linked dataset pairs. Our approach provides higher precision than SparseMap at the same recall level. This is because some dissimilar records in the original space become close in the Euclidean space due to information loss incurred during the embedding. By contrast, our approach does not suffer from false positives arising from the transformation.

7. RELATED WORK

Record linkage procedures have been refined over decades to integrate data in the face of dirty records [9]. At the same time, privacy concerns for sharing personal records has led to the development of private record linkage (PRL) protocols [3, 4, 16, 17, 23, 24, 28, 30]. PRL protocols tend to use two primary mechanisms to integrate data while protecting the privacy of the sensitive information: secure multiparty computation (SMC) and data transformation. SMC approaches offer accurate similarity evaluation to identify matching pairs with rigorous security guarantees. Various distance measures such as edit distance [2] and an approximation of Hamming distance [10] can be computed securely. Also, a recent fully homomorphic encryption scheme [12] enables secure computation of a wide range of functions. However, these methods are computationally intensive and do not scale for the integration of large data sets [6].

As an alternative, there are approaches to selectively reveal information through transformation [1, 16, 24, 25]. In [24], records are embedded into Euclidean space through a technique based on reference strings. Records are compared in the new space by a third party to identify matching pairs. As we illustrate in our experiments, this approach results in some false positives due to the approximations incurred during the embedding. In [30], the approach of [24] is extended to perform record linkage without the assistance of a third-party. In [23], public reference tables are used to form encoding for names. Specifically, a name is represented by its distance to all reference points. After encoding, the distance between pairs are estimated by a third-party. However, this approach leaks the distances between each record. As a result, the third party may exploit the residual information to disclose the names of the corresponding encodings (e.g., the distribution of distances may allow identification of rare names). In [1], a blocking-aware PRL algorithm was proposed. Their approach maps records into bit vectors according to term frequency / inverse document frequency scores of individual words in the records without considering the existence of errors. In [25], attributes of records are embedded into Bloom filters via a set of cryptographic hash functions. The encoded records are then compared via a set-based similarity measure. Although Bloom filter encoding tolerates errors in strings and offers relatively accurate linkage results, if not parameterized appropriately, it leaks uncontrolled information (e.g., the frequency distribution of individual attributes of the encoded records). In fact, under certain conditions, a cryptanalysis can be applied to retrieve the original records [21]. Additionally, [25] does not consider numeric attributes for record linkage.

In [16], a hybrid method for PRL that combines cryptographic and anonymization methods is proposed. This approach is based on anonymizing the datasets and linking record pairs through their anonymous versions. In [17], a similar hybrid approach is proposed which invoked differential privacy. However, the approach of [17] is designed only for numeric data. Hence, it cannot handle approximate string matching. Moreover, its differential privacy model does not discriminate against negative noise and suffers from excessive losses in identifying matching pairs.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a practical private record linkage protocol (PRL) with rigorous privacy guarantees that is resilient to dirty records in real world data sources. The proposed approach combines blocking with secure multiparty computation through a differentially private integration mechanism. To execute the protocol, public identifiers are clustered to form a common space representation for the parties contributing records to the linkage process. Based on these clusters, each party constructs blocks on their local datasets. The block construction enables a considerable reduction in the number of costly secure similarity computations that must be performed for each candidate record pair. After blocking, an efficient SMC technique is applied on the records of the same blocks to accurately identify matching pairs. In this setting, to integrate the blocking and SMC steps in a privacy preserving manner, we proposed a differentially private block release mechanism. Finally, we illustrated the success of the proposed scheme with an empirical analysis on a dataset that contains real personal identifiers.

In future work, we plan to extend the blocking scheme with fuzzy blocks (e.g., a record will be mapped to multiple blocks) to improve pair completeness (PC) of the blocking. Specifically, we'll use multiple attributes to generate multiple space representations for distinct parties. Fuzzy blocks will improve PC but at the same time it will lead to increase in the sensitivity of the differentially private block release. We'll formulate the balance between increase in PC and noise due to the differentially private block release analytically to find the best fuzzy block generation strategy.

9. ACKNOWLEDGMENTS

This work was partially supported by the Air Force Office of Scientific Research MURI-Grant FA-9550-08-1-0265 and Grant FA-9550-12-1-0082, National Institute of Health Grant 1R01LM009989, National Science Foundation (NSF) Grant Career-CNS-0845803, NSF Grants CNS-0964350, CNS-1016343, CNS-1111529 and CNS-1228198, Army Research Office Grant 58345-CS.

10. REFERENCES

[1] A. Al-Lawati, D. Lee, and P. McDaniel. Blocking-aware private record linkage. In *IQIS '05*, pages 59–68, 2005.

[2] M. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *WPES'03*, pages 39–44, 2003.

[3] T. Churces and P. Christen. Some methods for blindfolded record linkage. *Med. Informatics and Decision Making*, 4(9), 2004.

[4] C. Clifton, M. Kantarcioglu, A. Doan, G. Schadow, J. Vaidya, A. Elmagarmid, and D. Suciu. Privacy-preserving data integration and sharing. In *DMKD'04*, pages 19–26, 2004.

[5] N. C. V. R. Database. ftp://www.app.sboe.state.nc.us/data, 2011.

[6] E. Durham, Y. Xue, M. Kantarcioglu, and B. Malin. Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. *Information Fusion*, 13(4):245–249, 2012.

[7] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

[8] C. Dwork, K. McSherry, F.and Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[9] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: a survey. *TKDE*, 19(1):1–16, 2007.

[10] J. Feigenbaum, Y. Ishai, K. Nissim, M. Strauss, and R. Wright. Secure multiparty computation of approximations. *TALG*, 2(3):435–472, 2006.

[11] I. Fellegi and A. Sunter. A theory for record linkage. *JASA*, 64(328):1183–1210, 1969.

[12] C. Gentry. Fully homomorphic encryption using ideal lattices. In 41^{st} *STOC*, pages 169–178, 2009.

[13] S. Goldwasser and M. Bellare. *Lecture Notes on Cryptography, http://cseweb.ucsd.edu/ mihir/papers/gb.html*. 2008.

[14] M. Hernandez and S. Stolfo. Real-world data is dirty: data cleansing and the merge/purge problem. *DMKD*, 2(1):9–37, 1998.

[15] T. Herzog, F. Scheueren, and W. Winkler. *Data quality and record linkage techniques*. Springer, 2007.

[16] A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco. A hybrid approach to private record linkage. In *ICDE '08*, pages 496–505, 2008.

[17] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private record matching using differential privacy. In *EDBT '10*, pages 123–134, 2010.

[18] W. Jiang, M. Murugesan, C. Clifton, and L. Si. Similar document detection with limited information disclosure. In *ICDE'08*, pages 735–743, 2008.

[19] H. Kim and D. Lee. Harra: Fast iterative hashed record linkage for large-scale data collections. In *EDBT'10*, pages 525–536, 2010.

[20] M. Kuzu, M. Islam, and M. Kantarcioglu. Efficient similarity search over encrypted data. In *ICDE '12*, pages 1156–1167, 2012.

[21] M. Kuzu, M. Kantarcioglu, E. Durham, and B. Malin. A constraint satisfaction cryptanalysis of Bloom filters in private record linkage. In *PETS'11*, pages 226–245, 2011.

[22] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, pages 223–238, 1999.

[23] C. Pang, L. Gu, D. Hansen, and A. Maeder. Privacy-preserving fuzzy matching using a public reference table. *Intelligent Patient Management*, 189:71–89, 2009.

[24] M. Scannapieco, I. Figotin, E. Bertino, and A. Elmagarmid. Privacy preserving schema and data matching. In *SIGMOD '07*, pages 653–664, 2007.

[25] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using Bloom filters. *Med. Informatics and Decision Making*, 9(1):41, 2009.

[26] TypoGenerator. https://dbappserv.cis.upenn.edu/spell/, 2011.

[27] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *SIGKDD'02*, pages 639–644, 2002.

[28] V. Verykios, A. Karakasidis, and V. Mitrogiannis. Privacy preserving record linkage approaches. *IJDMMM*, 1:206–221, 2009.

[29] R. Xu and W. I. D. Survey of clustering algorithms. 16(3):645–678, 2005.

[30] M. Yakout, M. Atallah, and A. Elmagarmid. Efficient private record linkage. In *ICDE '09*, pages 1283–1286, 2009.