Privacy Preserving Group Linkage

Fengjun Li[1], Yuxin Chen[1], Bo Luo[1], Dongwon Lee[2], and Peng Liu[2,⋆]

[1] Department of EECS, University of Kansas
[2] College of IST, The Pennsylvania State University

Abstract. The problem of privacy preserving record linkage is to find the intersection of records from two parties, while not revealing any private records to each other. Recently, group linkage has been introduced to measure the similarity of groups of records [19]. When we extend the traditional privacy preserving record linkage methods to group linkage measurement, group membership privacy becomes vulnerable – record identity could be discovered from unlinked groups. In this paper, we introduce threshold privacy preserving group linkage (TPPGL) schemes, in which both parties only learn whether or not the groups are linked. Therefore, our approach is secure under *group membership inference attacks*. In experiments, we show that using the proposed TPPGL schemes, group membership privacy is well protected against inference attacks with a reasonable overhead.

Keywords: Group linkage, privacy, secure multi-party computation.

1 Introduction

Record linkage (RL), also known as the *merge-purge* [12] or *object identity* [24] problem, is one of the key tasks in data cleaning [10] and integration [9]. Its goal is to identify related records that are associated with the same entity from multiple databases. When we extend the concept of "records" to *"groups of records"*, it becomes the group linkage (GL) problem [19], which is to determine if two or more *groups of records* are associated with the same entity.

RL and GL problems occur frequently in inter-database operations, in which privacy is a major concern, especially in the presence of sensitive data. In both record and group linkage, data owners need to reveal identifiable attributes to others for record-level comparison. However, in many cases, data owners are not willing to disclose any attributes unless the records are proven to be related. Here we present two GL examples, in which private attributes should not be revealed.

Example 1: As an international coordination to combat against gang violence, law enforcement units from different countries collaborate and share information.

⋆ Bo Luo was partially supported by NSF OIA-1028098 and University of Kansas GRF-2301420. Dongwon Lee was partially supported by NSF DUE-0817376 and NSF DUE-0937891. Peng Liu was supported by AFOSR FA9550-07-1-0527 (MURI), ARO W911NF-09-1-0525 (MURI), NSF CNS-0905131, and NSF CNS-0916469.

J.B. Cushing, J. French, and S. Bowers (Eds.): SSDBM 2011, LNCS 6809, pp. 432–450, 2011.

Two countries will only share data when they confirm that they both possess information about the same gang group, which is represented as a set of records of gang members. Two gangs are regarded as the same when a large number of their members' records match. □

In this example, each party holds groups (i.e. gangs) of records (i.e. members) that are identified by primary keys (i.e. names). Two records "match" only if they have identical primary keys. This scenario represents *privacy preserving group linkage with exact matching* (PPGLE) problem, in which the similarity between two inter-group members takes value from $\{0, 1\}$.

Example 2: Two intelligence agencies (e.g. FBI and CIA) each obtains several pieces of intelligence documents. They would like to share the pieces if they are about the same case. Hence, the agencies need to verify that the similarity of their pieces are "very similar", in a way that does not reveal the document content in the verification process. □

In this case, each record (e.g. a document) is represented as a vector in a term space shared by both participants. Record-level similarity is measured by a similarity function $sim(\mathbf{r}, \mathbf{s})$, and takes value in $[0, 1]$. If the similarity between two group members is smaller than a preset *record-level cut-off*, it is considered as "noise" and set to 0. The group-level similarity is defined as a function of record-level similarity (e.g. $sum()$). This scenario represents *privacy preserving group linkage with approximate matching* (PPGLA) problem.

Privacy preserving *group* linkage (PPGL) extends privacy preserving *record* linkage (PPRL) such that participants hold groups instead of records. However, directly applying PPRL solutions to group linkage problems will suffer from *group membership inference attacks*. In such an attack, adversaries participate in the protocol with forged groups so that they can learn the group formation information of other parties, even though their groups are determined to be different in the end. To tackle this problem, we propose *threshold privacy preserving group linkage* (TPPGL) protocols for both exact matching (TPPGLE) and approximate matching (TPPGLA). The group similarity is no longer revealed to participating parties. Instead, only the result that the similarity is above or below a preset threshold is notified. In this way, private information about group membership is protected against inference attacks.

2 Problem Statement

Group linkage considers the problem of matching groups of records from multiple parities. For ease of presentation, hereafter, we use "Alice" and "Bob" to represent the two participants. In this scenario, Alice and Bob each holds a set of groups, identified as $\mathcal{R} = \{R_1, ..., R_u\}$ and $\mathcal{S} = \{S_1, ..., S_v\}$, respectively. Two groups are considered similar (i.e. linked) if and only if $SIM(R_i, S_j) \geq \theta$, where $SIM()$ is an arbitrary group similarity function and θ is a pre-negotiated threshold. In this paper, we follow the original group linkage definition [19] to

use Jaccard similarity [15] as the group-level similarity measurement (see Section 3.2 for details). In real-world applications, the number of elements in groups is usually small (e.g. tens), while the number of groups tends to be large (e.g. hundreds).

In the case of *exact matching*, a record (i.e., group member) is identified by a primary key, e.g. $R = \{r_1, ..., r_m\}$ and $S = \{s_1, ..., s_n\}$, where r_p and s_q are primary keys. Two records are regarded similar if and only if their primary keys are identical, i.e. $sim(r_p, s_q) = 1$, iff $r_i = s_j$. Note that, for ease of presentation, we use R instead of R_i to denote a group.

In the case of *approximate matching*, each record is represented as a vector in a shared vector space: $R = \{\mathbf{r}_1, ..., \mathbf{r}_m\}$, and $S = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$, where $\mathbf{r}_p$ and $\mathbf{s}_q$ are vectors. Two records are regarded similar if and only if their similarity is greater than a pre-negotiated record-level cut-off ρ, i.e., $sim(\mathbf{r}_p, \mathbf{s}_q) > \rho$. Here, $sim()$ is an arbitrary vector-space similarity function (e.g. cosine similarity or Euclidean distance). In this paper, we adopt the cosine similarity that is a popular similarity measure for text documents: $sim(\mathbf{r}_p, \mathbf{s}_q) = (\mathbf{r}_p \cdot \mathbf{s}_q)/(|\mathbf{r}_p||\mathbf{s}_q|)$.

Privacy preserving group linkage (PPGL): Both Alice and Bob follow a protocol to match two groups R from Alice and S from Bob. In the end, they learn $|R|$, $|S|$, and the group-level similarity $SIM(R, S)$, based on which they decide whether or not to share R and S. When exact matching or approximate matching is employed at the record level, the problem is further denoted as **PPGLE** or **PPGLA**, respectively.

The PPGL problem could be solved by existing privacy preserving set intersection protocols [1,4,11]. However, in the case where Bob holds a large number of groups with overlapping records, and Alice needs to check each of her groups against all Bob's groups, the existing solutions suffer from *group membership inference* problem, as shown in the following example.

Example 3. As shown in Figure 1 (a), Alice has a group of four records, where each record is identified by a primary key of last names. Bob has a set of three groups. Alice checks her group against Bob's three groups (using primary keys for exact matching), but fails to find a similar group (assume Jaccard similarity [15]

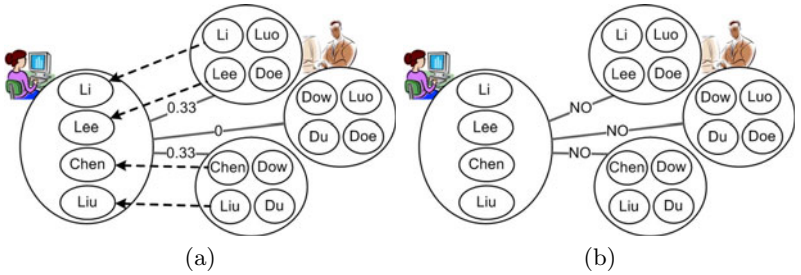


Fig. 1. (a) Privacy preserving group similarity with inference problem; (b) privacy preserving group linkage without inference problem

is used at group-level and θ is set to 0.5). In the three comparisons, both Alice and Bob learn the other's group sizes and the three similarities. Bob could easily infer the records in Alice's group via a simple derivation: (1) Alice's group should not have "Dow", "Du", "Luo", or "Doe" since its similarity with B_2 is 0; (2) Alice's group should have "Li" and "Lee" since its similarity with B_1 is 0.33 (which means the intersection size is 2); and (3) Alice's group should have "Chen" and "Liu" since its similarity with B_3 is also 0.33. Therefore, Alice's group is {`Li, Lee, Chen, Liu`} since the group size is known as 4. □

In this example, Bob does not learn the content of Alice's records (i.e. attributes other than the primary keys), since the comparisons are assumed privacy preserving. However, the *group membership privacy*, i.e. the identities of group members, is disclosed. In the example, Bob infers Alice's group members by providing partially overlapped groups. An adversary may intensionally manipulate his groups in a way that group-wise similarity is always below the threshold so that he does not need to share his groups, but he is able to infer the other party's group membership privacy. To tackle such a problem, we need to develop a secure protocol that only provides a verdict of "yes" or "no" for each group-wise comparison. Hence, Bob only learns three "no"s in the above example (as shown in Figure 2 (b)), and the inference attack becomes impossible.

Threshold privacy preserving group linkage (TPPGL): Alice and Bob negotiate a threshold θ, and then follow a protocol to match two groups R and S. In the end, they only learn $|R|$, $|S|$, and a boolean result B, where B=`true` when $SIM(R, S) \geq \theta$, and B=`false` otherwise. When exact matching or approximate matching is employed at the record level, the problem is further denoted as **TPPGLE** or **TPPGLA**, respectively.

3 Preliminaries

3.1 Cryptographic Primitives

The protocols proposed in the paper adopt two special classes of cryptography algorithms for secure computation: commutative and homomorphic encryption.

Commutative Encryption. An encryption algorithm has the *commutative* property when we encrypt a message twice (with two different keys) and the resulting ciphertext is independent of the order of encryptions. Mathematically, an encryption scheme $\mathsf{E}()$ is commutative if and only if, for any two keys e_1 and e_2 and any message m: (1) $\mathsf{E}_{e_1}(\mathsf{E}_{e_2}(m)) = \mathsf{E}_{e_2}(\mathsf{E}_{e_1}(m))$; (2) Encryption key e_i and its corresponding decryption key d_i are computable in polynomial time; and (3) $\mathsf{E}_{e_i}()$ has the same value range.

The commutative property applies to the decryption phase too. If a message is encrypted with keys e_1 and e_2, then it can be recovered by either decrypting the cipher using d_1, followed by decryption using d_2; or decrypting using d_2, followed by d_1. Here, d_i is the corresponding secret key of e_i. Several encryption algorithms are commutative, e.g. Pohlig-Hellman, ECC, etc. In this work, we adopt

SRA encryption scheme, which is essentially RSA, except that the encryption exponent e is kept private.

Homomorphic Encryption. Homomorphic encryption represents a group of semantically-secure public/private key encryption methods, in which certain algebraic operations on plaintexts can be performed with cipher. Mathematically, given a homomorphic encryption scheme $\mathsf{E}()$, ciphertexts $\mathsf{E}(x)$ and $\mathsf{E}(y)$, we are able to compute $\mathsf{E}(x \star y)$ without decryption, i.e. without knowing the plaintext or private keys. $\star$ represents an arithmetic operation such as addition or multiplication.

Well-known homomorphic encryption schemes include: RSA, El Gamal [5], Paillier [20], Naccache-Stern [18], Boneh-Goh-Nissim [2], and etc. The Paillier cryptosystem [20,21] is additively homomorphic; the El Gamal [5] cryptosystem is multiplicatively homomorphic; and the Boneh-Goh-Nissim cryptosystem approach [2] supports one multiplication between unlimited number of additions. A more recent approach provides full support of both addition and multiplication at higher computation costs [6,25]. We omit further mathematical details in this paper, since they are out of our scope.

3.2 Related Work

The problem of privacy preserving group linkage originates from secure two-party computation and group linkage (which succeeds record linkage). We briefly summarize the literature in these areas.

Group linkage. The record linkage or merge-purge problem has been intensively studied in database, data mining, and statistics communities [27,3]. Group linkage [19] extends the scenario to take groups of records into consideration. Group-wise similarity [19] is calculated based on record-level similarity. When exact matching is enforced at the record level, *Jaccard similarity*[15] is employed at the group level: similarity of two groups (R and S) is defined as: $SIM(R,S) = |R \cap S|/|R \cup S|$. When approximate matching is applied at the record level, *bipartite matching similarity* is employed at the group level [19]. For two groups of records $R = \{\mathbf{r}_1, ..., \mathbf{r}_m\}$ and $S = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$, $BM_{sim,\rho}$ is the normalized weight of M:

$$BM_{sim,\rho}(S,R) = \Big(\sum_{(\mathbf{r}_i,\mathbf{s}_j)\in M} sim(\mathbf{r}_i,\mathbf{s}_j)\Big) \;/\; (|R| + |S| - |M|)$$

where M indicates the maximum weight matching in the bipartite graph ($\mathcal{N} = R \cup S, \mathcal{E} = R \times S$). It contains all the edges whose weight is greater than ρ, i.e. $(\mathbf{r}_i, \mathbf{s}_j) \in M$ iff. $sim(\mathbf{r}_i, \mathbf{s}_j) \geq \rho$.

Privacy preserving record linkage. The original problem of *secure two/multi-party computation* was introduced in [28]. In this problem, multiple parties compute the value of a public function on private variables, without revealing the values of the variables to each other. Zero-knowledge proof [8] addresses the

Protocol 1. AES approach for set intersection [1]

Data: Alice has a group $R = \{r_1, ..., r_m\}$; and Bob has a group $S = \{s_1, ..., s_n\}$.
Result: They both learn the size of intersection: $|R \cap S|$, and nothing else.

1: Both Alice and Bob apply hash function to their group elements to obtain: $h(R) = \{h(r_1), ..., h(r_m)\}$ and $h(S) = \{h(s_1), ..., h(s_n)\}$.
2: Both Alice and Bob encrypt their hashed group elements to obtain: $\mathsf{E}_r(h(R)) = \{\mathsf{E}_r(h(r_1)), ..., \mathsf{E}_r(h(r_m))\}$ and $\mathsf{E}_s(h(S)) = \{\mathsf{E}_s(h(s_1)), ..., \mathsf{E}_s(h(s_n))\}$.
3: Alice and Bob exchange their group, with group elements reordered.
4: Bob encrypts what he got from Alice to obtain: $\mathsf{E}_s(\mathsf{E}_r(h(\mathcal{R}))) = \{\mathsf{E}_s(\mathsf{E}_r(h(r_1))), ..., \mathsf{E}_s(\mathsf{E}_r(h(r_n)))\}$, and return to Alice.
5: Alice encrypts what she got from Bob in Step 3, to obtain: $\mathsf{E}_r(\mathsf{E}_s(h(S)))$.
6: Alice finds out the size of intersection of the encrypted groups, $\mathsf{E}_s(\mathsf{E}_r(h(S)))$ (step 4) and $\mathsf{E}_r(\mathsf{E}_s(h(R)))$ (step 5), and shares with Bob.

problem of proving the veracity of a statement to other parties without revealing anything else. They are the earliest ancestors of privacy preserving multi-party computing. Privacy preserving record linkage with exact matching is very similar to privacy preserving set intersection: to identify the intersection of two sets of records without revealing private records. Surveys could be found at [26,11]. Among the more popular approaches, solutions based on homomorphic encryption (e.g. [4,16]) or commutative encryption (e.g. [1]) require higher computational overhead. Sanitization-based approaches (e.g. [14]) modify sensitive data so that they are not identifiable among others, but they are not suitable when there are a small number of records or the participants require perfect privacy protection. A hybrid approach [13] combines sanitization and crypto-based techniques to provide a balance among privacy, accuracy and computation (cost). In the context of approximate matching, there have been proposals on privacy preserving similar document detection (e.g. [22,17]).

3.3 Privacy Preserving Group Linkage: Baseline Solutions

Privacy preserving record linkage protocols match related records shared by two parties, without revealing any private records. This requires encrypting records so that computations (or comparisons) can be conducted on the ciphertexts. Agrawal et al. proposed a commutative encryption based solution in [1], which we refer as the AES protocol, and Freedman et al. presented a homomorphic encryption based scheme in [4], which we refer as the FNP protocol. We briefly introduce the protocols in Protocol 1 and 2. For more details, please refer to their papers [4,1].

These protocols serve as the baseline approaches for PPGLE, in which group-wise similarities are revealed to participants, but record information (for both shared and private records) is kept private. However, as we have described in Section 2, such solutions suffer from group membership inference attacks.

Protocol 2. FNP approach for set intersection size [4]

Data: Alice has a group $R = \{r_1, ..., r_m\}$; and Bob has a group $S = \{s_1, ..., s_n\}$.
Result: They both learn the size of intersection: $|R \cap S|$, and nothing else.

1: Alice creates keys for homomorphic encryption and publishes her public key.
2: Alice constructs $R(x) = \prod(x - r_i)$, and computes all the coefficients α_u that $R(x) = \sum_{u=0}^{u=m} \alpha_u x^u$. Therefore, the m degree polynomial $R(x)$ has roots $\{r_1, ..., r_m\}$.
3: Alice encrypts the coefficients and sends them $(\{\mathsf{E}(\alpha_0), \mathsf{E}(\alpha_1), ..., \mathsf{E}(\alpha_m))$ to Bob.
4: For each s_j, Bob evaluates the polynomial (without decryption) to get $\mathsf{E}(R(s_j))$.
5: Bob chooses a random value γ, and a pre-negotiated spacial value ν. For each $\mathsf{E}(R(s_j))$, he further computes $\mathsf{E}(\gamma * R(s_j) + \nu)$.
6: Bob permutes his set of $\mathsf{E}(\gamma * R(s_j) + \nu)$, and return them to Alice.
7: Alice decrypts all $\mathsf{E}(\gamma * R(s_j) + \nu)$. For each $s_j \in S \cap R$, she gets ν; otherwise, she gets a random value. Alice counts the number of ν values, and output.

4 TPPGL with Exact Matching

4.1 TPPGLE Using Commutative Encryption

In threshold privacy preserving group linkage, Alice and Bob first negotiate a group-level threshold θ: two groups are regarded similar if and only if $SIM(R, S) \geq \theta$. With exact matching at record-level, $sim(r_i, s_j) \in \{0, 1\}$. Let k be the minimum number of identical records from two groups for them to be linked, we have: $SIM(R, S) = \dfrac{k}{|R| + |S| - k} \geq \theta$. Note that we employ Jaccard similarity at group level.

If $|R| = m, |S| = n$, we have $k = \lceil (m + n)\theta/(1 + \theta) \rceil$; i.e. k is the smallest integer that is greater than or equal to $(m+n)\theta/(1+\theta)$. Therefore, the TPPGLE problem is to securely compare the actual intersection size ($|R \cap S|$) with k in a way that $|R \cap S|$ **should not be revealed to either Alice or Bob**[1]. Please note that although group sizes are revealed, it is acceptable since they could not be used to infer record identity. Similarly, privacy preserving record linkage solutions also share the number of records among participants.

To extend the AES scheme to tackle the TPPGLE problem, we enumerate all k-combinations of Alice's and Bob's elements and compare the k-combinations in privacy preserving manner. If at least one of the pairwise comparisons of k-combinations yields a positive result, we conclude that k or more elements from two groups match, so that the Jaccard similarity of the two groups has reached the threshold, and Alice and Bob should share the groups. The detailed process is shown in Protocol 3.

Figure 2 gives an example of Protocol 3. In this example, Alice and Bob each hold groups of records identified by names. Figure 2 (a) shows Alice's group of four records and Bob's group of three records. A threshold $\theta = 0.7$ is pre-negotiated. In step 1, both Alice and Bob get $k = \lceil 0.7(7)/(1 + 0.7) \rceil = 3$. In step

[1] This requirement makes the problem different from the well-know Yao's millionaire problem [28].

Protocol 3. K-combination approach for TPPGLE

Data: Alice has a group $R = \{r_1, ..., r_m\}$; and Bob has a group $S = \{s_1, ..., s_n\}$. They negotiate a similarity threshold θ.

Result: Both Alice and Bob learn whether or not the similarity between R and S is greater than θ, i.e. if $SIM(R, S) > \theta$, and nothing else; especially, not $SIM(R, S)$.

1: Alice and Bob both compute $k = \lceil (m+n)\theta/(1+\theta) \rceil$, i.e. the smallest integer that is not less than $(m+n)\theta/(1+\theta)$.
2: Alice and Bob each gets all the k-combinations of her/his own elements. There are C_k^m k-combinations from Alice and C_k^n k-combinations from Bob.
3: For each k-combination, sort the elements using a pre-negotiated order, and serialize them into a string, with a special separator between elements.
4: Both Alice and Bob follow the AES approach (Protocol 1) to find the intersection of the k-combinations. If at least one k-combination is found in the intersection, the two groups are matched, i.e. $SIM(R, S)$ is guaranteed to be greater than θ.

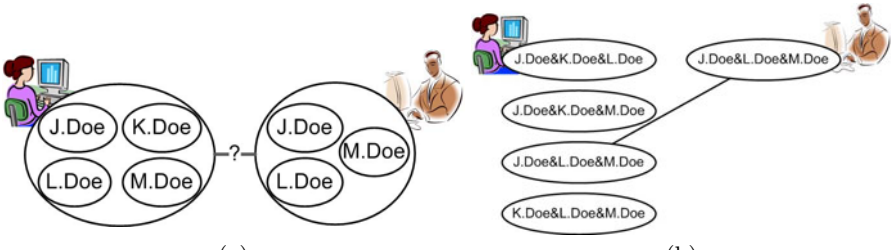


Fig. 2. (a) Alice and Bob's groups; (b) privacy preserving set intersection of k-combinations extracted from groups

2, Alice gets C_3^4 3-combinations from her records and Bob gets C_3^3 3-combinations from his records. In step 3, Alice sorts the elements in each 3-combination in ascending order, serializes the primary keys into a string, with "&" as the separator. As shown in Figure 2 (b), Alice and Bob continue to use AES approach to find the intersections of the strings, in a privacy preserving manner. In this example, one intersection is found, which means the two groups are considered to be matched at the threshold of 0.7. On the other hand, if we replace Alice's record "J.Doe" with another record "Z.Doe", follow the above procedures, then, none of the strings serialized from 3-combinations would match. In this case, two groups are not linked, and both Alice and Bob learn nothing about other's group.

In this approach, we avoid homomorphic encryption, which requires heavy computation. However, when Alice generates C_k^m k-combinations and Bob generates C_k^n k-combinations, the value of C_k^m and C_k^n could be too large to manipulate. Therefore, this approach is preferable when $k = \lceil (m+n)\theta/(1+\theta) \rceil$ is (1) very small, or (2) very close to m and n. In real-world applications, k is usually close to m and n.

4.2 TPPGLE Using Homomorphic Encryption

In FNP approach, Alice and Bob pre-negotiate a special value ν, which represents a matching record. After decryption, Alice counts the number of ν values, which represents the number of records in $R \cap S$. In TPPGLE, this number should not be revealed to either party. Instead, they should only learn a Boolean value: $|R \cap S| > k$. To tackle this problem, we modify FNP approach starting from step 6. Before permuting $\mathsf{Enc}(\gamma * R(s_j) + \nu)$, Bob injects a random number (k_b) of $\mathsf{Enc}(\nu)$ elements into the result set. We assume there is $k' = |R \cap S|$, which generates k' number of $\mathsf{Enc}(\nu)$ elements in the original $\mathsf{Enc}(\gamma * R(s_j) + \nu)$ set. After the random injection, Alice decrypts the polluted set to obtain $(k_b + k')$ ν values. She has no knowledge about either k_b or k', as long as k_b is selected from a good range.

Now the problem is converted to Yao's Millionaire Problem: Alice knows k_b+k' while Bob knows k_b+k (the new threshold), and they want to compare two values without leaking them to each other. In our settings, $k_b + k' \ll N, k_b + k \ll N^2$. We may assume that the product of $(k'-k)$ and a random number r' is much less than N. In our solition, Bob first generates $\mathsf{Enc}(k'-k)$ from $\mathsf{Enc}(k_b+k')$ (obtained from Alice) and $\mathsf{Enc}(k_b + k)$. Hence, we are to find out whether $k' - k > 0$ or not, without revealing the actual value. Bob further randomizes the result with two positive random numbers $\gamma' \ll N$ and $\nu' < \gamma'$, obtaining $\mathsf{Enc}(\gamma' \times (k' - k) + \nu')$. Then Alice gets the cipher from Bob and decrypts it. Based on previous assumption, we may infer that $(\gamma' \times (k' - k) + \nu') > 0$ iff. $(k' - k) > 0$, and vice versa. Meanwhile, due to the cryptographic properties of Paillier cryptosystem, the decryption result should be the least positive residues of plain text modulus N. Hereby, if $\gamma' \times (k' - k) + \nu' < N/2$, we have $k' - k > 0$ and thus the two groups are linked. Otherwise, if $N/2 < \gamma' \times (k' - k) + \nu' < N$, the two groups are not linked.

5 TPPGL with Approximate Matching

In previous section, group members (records) are identified by primary keys. Therefore, two records are either "identical" or "different". In this section, we consider the problem with approximate matching. In this scenario, Alice and Bob pre-negotiate a vector space, and represent their records as vectors in this space. Since our research is more focused on group-level linkage, we adopt a simple cosine similarity function, which employs private scalar product [7], for document-level vector-space similarity.

First, in Protocol 5, we revisit the privacy preserving inner-product (scalar product) approach presented in [7]. In the protocol, z represents the dimensionality of the vector space, while μ denotes the normalized modulus of space vectors.

[2] k is no larger than the group size. In our assumptions, typical group size is small (e.g. tens). On the other hand, in our experiments, N is the product of two 256-bit prime numbers.

Protocol 4. Homomorphic encryption approach for TPPGLE

Data: Alice has a group $R = \{r_1, ..., r_m\}$; and Bob has a group $S = \{s_1, ..., s_n\}$. They negotiate a similarity threshold θ.

Result: Both Alice and Bob learn whether or not the similarity between R and S is greater than θ, i.e. if $SIM(R, S) > \theta$, and nothing else; especially, not $SIM(R, S)$.

1: Alice creates keys for homomorphic encryption and publishes her public key.
2: Alice constructs $R(x) = \prod(x - r_i)$, and computes all the coefficients α_u that $R(x) = \sum_{u=0}^{u=m} \alpha_u x^u$. Therefore, the m degree polynomial $R(x)$ has roots $\{r_1, ..., r_m\}$.
3: Alice encrypts the coefficients and sends them $(\{\mathsf{E}(\alpha_0), \mathsf{E}(\alpha_1), ..., \mathsf{E}(\alpha_m))$ to Bob.
4: For each s_j, Bob evaluates the polynomial (without decryption) to get $\mathsf{E}(R(s_j))$.
5: Bob chooses a random value γ, and a pre-negotiated spacial value ν. For each $\mathsf{E}(R(s_j))$, he further computes $\mathsf{E}(\gamma * R(s_j) + \nu)$.
6: Bob gets a random number k_b. He injects k_b number of $\mathsf{E}(\nu)$ values into the set he obtained from the previous step. Meanwhile, he also injects random number of random values into this set.
7: Bob permutes his polluted set of $\mathsf{Enc}(\gamma * R(s_j) + \nu)$, and returns them to Alice.
8: Alice decrypts all items in the polluted set. She then count number of ν values.
9: Assume $k' = |R \cap S|$, Alice now knows $k_b + k'$, but not k_b; Bob knows k_b, and thus $k_b + k$. Neither of them knows k'.
10: Alice encrypts $k_b + k'$, and sends it to Bob.
11: Bob gets $\mathsf{E}(k_b + k')$. With the homomorphic properties of $\mathsf{E}()$, he calculates $\mathsf{E}((k_b + k') - (k_b + k)) = \mathsf{E}(k' - k)$.
12: Bob creates two random numbers $\gamma' \ll N$ and $\nu' < \gamma'$. Bob randomizes $\mathsf{E}(k' - k)$ to $\mathsf{E}(\gamma' \times (k' - k) + \nu')$.
13: Bob return $\mathsf{Enc}(\gamma' \times (k' - k) + \nu')$ to Alice. Alice decrypts it to m, output "Yes" if $m < N/2$, or "No" if $m > N/2$.

We assume Alice has a group: $R = \{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_m\}$, and Bob has a group $S = \{\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_n\}$. In simple PPGLA, we conduct pairwise comparison between Alice's and Bob's vectors, and all $sim(\mathbf{r}_i, \mathbf{s}_j)$ are counted towards group similarity. Being simple in calculation, however, this approach does not provide best result, since many $sim(\mathbf{r}_i, \mathbf{s}_j)$ values are very small that they should be considered as "noise". Therefore, a better solution is to have a record-level "cut-off" ρ such that: edge is created in the bipartite graph iff. the similarity between two vertexes is larger than ρ (i.e. $sim(\mathbf{r}_i, \mathbf{s}_j) > \rho$). On the other hand, also to eliminate noises, we only consider an unlabeled bipartite graph – when two records are linked, we use "1", instead of $sim(\mathbf{r}_i, \mathbf{s}_j)$ in group-level similarities. With a binary bipartite graph, group-wise similarity becomes: $BM_{sim,\rho}(R, S) = \dfrac{k}{|R| + |S| - k}$, where $k = |M|$. To get $BM_{sim,\rho}(R, S) > \theta$, we need to have: $k > (m+n)\theta/(1+\theta)$.

Therefore, Alice and Bob need to pre-compute k_{min} based on θ, then securely compute k, and compare k with k_{min}. However, this approach is again flawed – an advisory could break the protocol by faking his groups. Let us assume that Alice and Bob each has a group of three members, while only $\mathbf{r}_1$ and $\mathbf{s}_1$ match $(sim(\mathbf{r}_1, \mathbf{s}_1) > \rho)$. Bob could fake a group with (repeated) members: $\{\mathbf{s}_1, \mathbf{s}'_1, \mathbf{s}''_1\}$,

Protocol 5. Privacy preserving inner-product [7].

Data: In a shared vector space $\mathbb{Z}_\mu^z, \mu < \sqrt{N/2z}$, Alice has her vector $\mathbf{r}$; and Bob has his vector $\mathbf{s}$.

Result: Alice and Bob both learn the inner-product $\mathbf{r} \cdot \mathbf{s}$.

1: Alice creates keys for homomorphic encryption and publishes her public key.
2: Alice encrypts each r_i in $\mathbf{r} = [r_1, ..., r_z]$ to obtain $\mathsf{Enc}(\mathbf{r}) = [\mathsf{E}(r_1), \mathsf{E}(r_2), ..., \mathsf{E}(r_z)]$.
3: Alice sends $\mathsf{E}(\mathbf{r})$ to Bob.
4: With the homomorphic properties of $\mathsf{E}()$, Bob computes the inner-product (without decryption): $\mathsf{E}(\mathbf{r} \cdot \mathbf{s}) = \mathsf{E}(r_1 s_1 + r_2 s_2 + ... + r_z s_z)$.
5: Bob sends $\mathsf{E}(\mathbf{r} \cdot \mathbf{s})$ back to Alice, Alice decrypts and publishes the result.

in which $\mathbf{s}'_i$ is a slightly modified version of $\mathbf{s}_i$. In this way, the manufactured group is highly likely to be linked with R. To tackle this problem, we measure the "degree of participation" from Alice and Bob, instead of using the total number of linked records. In other words, we count the number of elements from Alice that are linked to at least one element from Bob (m'), and vice versa. If we obtain m' and n' from the count, we then compare $min(m', n')$ with k_{min} to make the decision.

To implement such operations in a privacy preserving manner, we present TPPGLA in Protocol 6 (on the last page). In the protocol, Alice and Bob will use an encrypted similarity matrix M to store the intermediate results. The content of M should be private throughout the group linkage procedure. Bob first generates $\mathsf{E}(\mathbf{r}_i \cdot \mathbf{s}_j)$ with the private-preserving scalar product protocol, and subtract them by the record-level cut-off ρ. In the matrix M, each positive value (in plaintext) indicate a link at the record level (or an edge in the bipartite marching graph). If a row i has at least one positive value, it indicates that Alice's record $\mathbf{s}_i$ has participated in the linkage (i.e. linked with at least one record from Bob). To measure m' is to count number of rows that have at least one positive value, and to measure n' is to count number of columns that has at least one positive element.

To conduct such operations securely, Bob randomizes each pairwise record similarity into a encrypted "Boolean" value, with meaningful information only in the sign digit of the plain text. Before sending the cipher back to Alice, Bob injects two groups of positive and negative values with random sizes into each row and column, expanding the size of M into a larger range. If Alice counts all the positive values in row i, of M, she cannot infer whether $\mathbf{r}_i$ shares any similar records in Bob's set (i.e. the number of links between $\mathbf{r}_i$ and Bob's items, c_{ri}), if she doesn't how many positive values Bob has injected.

Further more, to protect m' and n' from been learned by either party, Bob performs another injection-permutation operation after Alice returns (encrypted) sum of each row. He injects c_{br} non-zero values into the set of c_{ri}, and make sure that Alice can only learn $m' + c_{br}$ instead of m'. Similarly, Alice can obtain $n' + c_{br}$, where n' denotes the number of shared items from Bob.

In approximate matching, if Alice shares m' records with Bob, and Bob shares n' records with Alice, the maximum bipartite matching cardinality is

Protocol 6. *TPPGLA* with record-level cut-off

Data: Alice has a group $R = \{\mathbf{r}_1, ..., \mathbf{r}_m\}$; Bob has a group $S = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$. They negotiate a record-level cut-off threshold ρ and a similarity threshold θ.

Result: Alice and Bob both learn if similarity between $\mathcal{R}$ and $\mathcal{S}$ is greater than θ, i.e. if $BM_{sim}(\mathcal{R}, \mathcal{S}) > \theta$, and nothing else; especially, not $BM_{sim,\rho}(\mathcal{R}, \mathcal{S})$.

1: Alice creates keys for homomorphic encryption and publishes her public key.
2: Alice and Bob negotiate a shared space $\mathbb{Z}_\mu^z$. They represent their vectors $\mathbf{r}_i$ and $\mathbf{s}_j$ in the space. All vectors are normalized to $\mu < \sqrt{N/2z}$.
3: Alice and Bob both compute $k = \lceil (m+n)\theta/(\rho+\theta) \rceil$.
4: For each pair $\mathbf{r}_i, \mathbf{s}_j$, they follow protocol 5 to compute $\mathsf{E}(\mathbf{r}_i \cdot \mathbf{s}_j)$. Instead of sending $\mathsf{E}(\mathbf{r}_i \cdot \mathbf{s}_j)$ to Alice, Bob chooses a random value $\gamma > 0$ to compute $\mathsf{E}(\gamma * (\mathbf{r}_i \cdot \mathbf{s}_j - \rho))$. The result set forms a $m \times n$ matrix $M = (\mathsf{E}(\gamma * (\mathbf{r}_i \cdot \mathbf{s}_j - \rho))_{m \times n}$.
5: Bob creates two random vectors $\boldsymbol{c}_{b+}$ and $\boldsymbol{c}_{b-}$, where $\boldsymbol{c}_{b+} = (c_{b1+}, c_{b2+}, ..., c_{bm+})$ and $\boldsymbol{c}_{b-} = (c_b - c_{b1+}, c_b - c_{b2+}, ..., c_b - c_{bm+})$. For each row of M, Bob injected c_{bi+} encrypted random positive values and c_{bi-} encrypted random negative values and gets $M_r = [M]_{m \times (n+c_b)}$. He then permutes each row of M_r and sends it to Alice.
6: Alice decrypts M_r into V_r. She counts the number of $\nu < N/2$ for each row, and obtains $c_{ri} + c_{bi+}$, where c_{ri} denotes the number of records in $\mathcal{S}$ that are supposed to be similar with $\mathbf{r}_i$. Alice now knows $c_{ri} + c_{bi+}$ and Bob knows c_{bi+}. Neither of them knows whether $\mathbf{r}_i$ is similar with any record in S.
7: Alice encrypts $[c_{r1} + c_{b1+}, c_{r2} + c_{b2+}, ..., c_{rm} + c_{bm+}]$ to $[\mathsf{E}(c_{r1} + c_{b1+}), \mathsf{E}(c_{r2} + c_{b2+}), ..., \mathsf{E}(c_{rm} + c_{bm+})]$ and sends them to Bob.
8: Bob creates two positive random numbers γ and υ. He randomizes $\mathsf{E}(c_{ri} + c_{bi+} - c_{bi+})$ and gets $[\mathsf{Enc}(\gamma * c_{r1} + \nu), \mathsf{E}(\gamma * c_{r2} + \nu), ...\mathsf{Enc}(\gamma * c_{rm} + \nu)]$.
9: Bob creates a random number c_{br} of different random integers $0 < d_{inject} < m$. He injects c_{br} number of $\mathsf{Enc}(d_{inject})$ values into $[\mathsf{E}(\gamma * c_{r1} + \upsilon), \mathsf{E}(\gamma * c_{r2} + \nu), ...\mathsf{Enc}(\gamma * c_{rm} + \nu)]$, permutes the set and sends the result to Alice.
10: Alice decrypts all items in the polluted set. Assume $\mathcal{R}_{m'}$ is the largest subset of $\mathcal{R}_m$, $\forall \mathbf{r}_i \in \mathcal{R}_{m'}$, $\exists \mathbf{s}_j \in \mathcal{S}$, s.t. $sim(\mathbf{r}_i, \mathbf{s}_j) > \rho$. She then count number of non-zero values and gets $m' + c_{br}$.
11: Similarly, Alice learns $n' + c_{br}$ if we conduct the above operations in columns.
12: Now Alice knows $m' + c_{br}$ and $n' + c_{br}$, and Bob knows $k + c_{br}$. They can proceed with protocol 4 from step 9 to compare $min(m', n')$ with k. If intersection threshold k is smaller than $min(m', n')$, the $SIM(R, S)$ is guaranteed to be greater than θ.

$min(m', n')$. Alice and Bob both compute the group intersection threshold k; now Alice knows $\{m' + c_{br}, n' + c_{br}\}$ and Bob knows $k' + c_{br}$, and they want to learn if $k' < min(m', n')$. They can follow Protocol 4 from step 9 to get the final decision.

6 Security Analysis

6.1 Attacker Models

The goal of the proposed TPPGL protocols is to guarantee that (1) in the protocol, each party learns only the fact whether the groups are similar or not; and (2) no content or similarity measurement at the record level is disclosed

to any party; no similarity measurement at the group-level (other than (1)) is disclosed to any party. Please note that in privacy preserving record linkage, it is convention that numbers of records from all participants are revealed. In our scenario, it is also acceptable that both parties disclose the sizes of their groups. Unlike group similarity information, group size information cannot be used to infer record identities.

In secure two-party computation problems, communication between the parties is usually assumed to be authenticated and encrypted. Therefore, our protocol is secure against *outsider adversaries* that passively monitor the network traffic but have no access to the inputs or the outputs. Hence, we further consider two types of insider adversaries, *semi-honest* (a.k.a. *honest-but-curious*) and *augmented semi-honest* adversaries, in our attacker model: (1). Semi-honest model describes a passive insider attack: both parties are assumed to properly follow protocol (so that they are "honest"); meanwhile, each party keeps all the accessible intermediate results and outputs, and tries to infer private data (so that they are "curious"). (2). In augmented semi-honest attacker model, the adversary can further change the input and output of one party, without tampering the protocol, to affect the views of the others'.

Due to space limitations, we only evaluate the **correctness** and **security** of Protocol 3, i.e. TPPGL with commutative encryption. With the same methodology, we can extend our proofs to all other protocols presented in the paper.

6.2 TPPGLE Using Commutative Encryption

First, we classify the two parties as *client* and *server* based on their roles in the protocol. The client (e.g. Alice) initiates the secure computation and gets the output, and the server (e.g. Bob) responds to the inputs from the client. The protocol is correct if it evaluates the similarity function with high probability.

Statement 1. *In Protocol 3, assuming there are no hash collisions, the client learns a Boolean output, where 1 for* $SIM(R,S) > \theta$*, and 0 otherwise.*
Proof. For any $k \leq min(|R|,|S|)$, assume the hash function h has no collisions on $R^k \cup S^k$. Since E_s and E_r are bijective and commutative, we have
$\upsilon \in R^k \cap S^k$ iff $\upsilon \in R^k$ and $\mathsf{E}_r(\mathsf{E}_s(h(\upsilon))) = \mathsf{E}_s(\mathsf{E}_r(h(\upsilon)))$,
which means the same concatenated set of elements is constructed by both the client and the server. Since k is calculated from θ, we have $SIM(R,S) > \theta$ iff $\exists \upsilon \in R^k \cap S^k$. □

The security of the protocol is to preserve the privacy of the data of both client and server. Then, we have

Statement 2. *TPPGLE-Commutative is secure if both parties are semi-honest or augmented semi-honest. From the protocol, the client learns the Boolean output and the size* $|S|$*, and the server only learns the size* $|R|$*.*

Proof. We use the similar proof methodology as in [1]: it assumes a simulation using the knowledge that the client (and the server) is supposed to have according

to the protocol, and the client (and the server) should not be able to distinguish the simulated view and the real view.

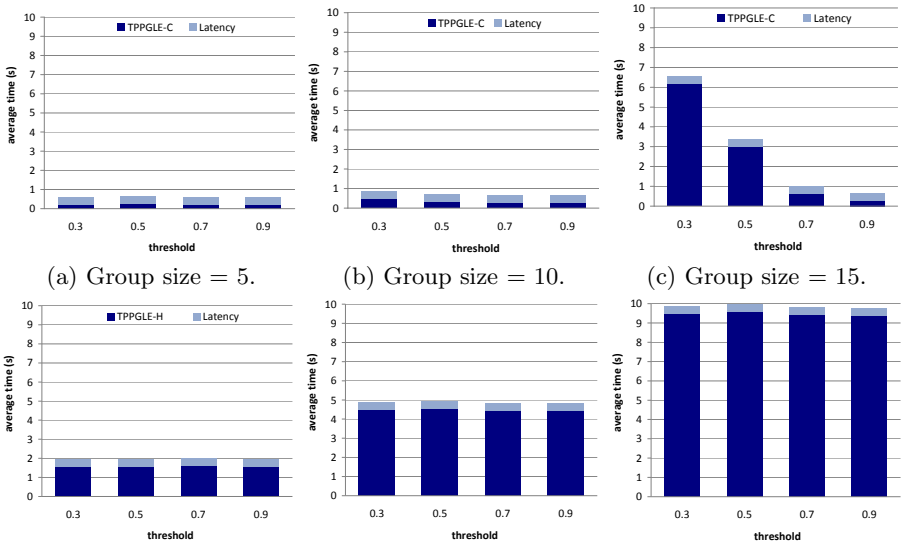
First, let us construct the simulator for the server. The server receives no output from the protocol, but $C_k^{|R|}$ encrypted messages from the client at step 4. Each message is the hash of the concatenation of k elements from set R, encrypted by commutative key E_r. The simulator generates k random values $z_i \in \digamma$, where $\digamma$ is the message space, and then concatenates them in a random sequence. Assume the hash $h(||z_1||...||z_k||)$ is uniformly distributed ("||" denotes concatenation), the real view and the simulated view for the server are indistinguishable.

Then, let us construct the simulator for the client. The simulator will use R, $|S|$, and $R^k \cap S^k$. To simulate the view for the client, the simulator selects a commutative key E_h to encrypt $||z_1||...||z_k||$ for $z_i \in R \cap S, 1 \leq i \leq k$. Then the simulator generates $|S| - |S \cap R|$ random k-concatenations, and encrypts them with E_h. In real view, the $|S|$ concatenations are all encrypted by E_s. Since E_h and E_s are randomly chosen from the same key space, their distributions are identical. For the client, the real view and the simulated view are indistinguishable. □

7 Experiments

We perform our experiments on three data sets, which were adopted in [23]. As summarized in Table 1, two data sets, *co-author network* (shortly AN) and *paper citation network* (shortly CN), are extracted from academia search system Arnetminer, and the last one, *movie network* (shortly MN), is crawled from Wikipedia category "English-language films". AN represents author names as vertices and the coauthor relationships as edges, while in CN, the vertices are a set of 2,329,760 papers and the edges denote the citation relationships between the papers. Since both AN and CN are homogeneous networks, we treat each 1-neighborhood subgraph (e.g. an author and all the co-authors) as a group, and use author name and citation name as key attributes for exact matching. MN is a heterogeneous network with 142,426 relationships between the heterogeneous nodes of films, director, actors, and writers. In our experiments, we treat a heterogeneous subnet (of a selected number of nodes) as a "group", and extract the label from each node to form the content of "records". Textual similarity (e.g. cosine similarity with TF/IDF weighting) between two labels is calculated as pairwise record-level similarity.

Then, we quantitatively evaluate the performance of the proposed TPPGL protocols (Protocol 3, 4 and 6) on the three data sets. Since our focus is on the viability and performance of these approaches in privacy preserving group linkage, we measure the *end-to-end execution time* under different group sizes and thresholds θ to assess the efficiency of each protocol. To meet the computational requirements of the cryptographic operations, we implement our own Big Integer class in C# under .NET framework 3.5.



(a) Group size = 5. (b) Group size = 10. (c) Group size = 15.

(d) Group size = 5. (e) Group size = 10. (f) Group size = 15.

Fig. 3. Experimental results of TPPGLE

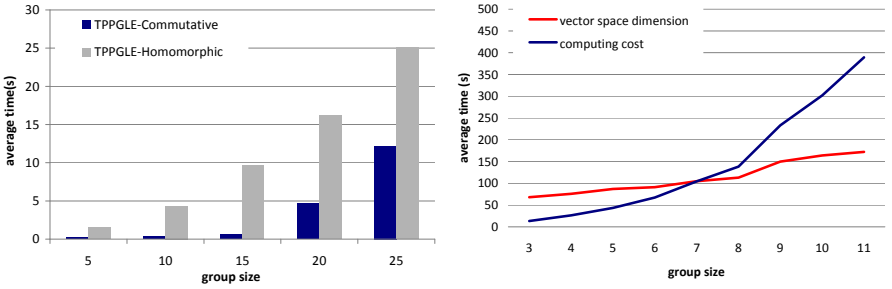
Table 1. A summary of three data sets

Data Set	Key	Record
AN	Author name	Authors and coauthors.
CN	Paper name	Paper and citations.
MN	-	Attributes (labels) of actors, writers, singers, etc.

7.1 TPPGLE

To evaluate the performance of TPPGLE, we first generate synthetic groups from AN and CN data sets. To form a group, we randomly pick a seed node, and follow its edges to add more members to the group. For instance, when a seed (3, J. Doe(15), 203) is selected, the next node 203 should be added to the group. We evaluate the protocol under different group sizes (e.g. each group has 5, 10, and 15 records) and different thresholds (e.g. $\theta \in \{0.3, 0.5, 0.7, 0.9\}$). For each θ, we generate 50 pairs of linked groups, and 50 pairs of unrelated groups.

Following the TPPGLE protocol using commutative encryption, we first hash each record (of Alice and Bob) into a 160-bit value, and then encrypt it with a commutative encryption function. Here, we adopt the famous SRA scheme. The average end-to-end execution time for Protocol 3 under different group sizes are shown in Figures 3(a)-(c). In each figure, the lower portion (denoted as TPPGLE-C) of the tacked bars represents the average execution time under different thresholds, and the upper portion represents network latency delay, which is estimated as the average one-hop network latency (i.e. 100ms) times the



(a)TPPGLE with difference group sizes; (b) TPPGLA with difference group sizes.

Fig. 4. Computation cost of TPPGL protocols

rounds for data exchange between Alice and Bob. From the results, we see that the performance of Protocol 3 highly depends on the preset similarity threshold, especially when group size becomes large. It is easy to understand: a larger similarity threshold θ, which means k is closer to the group size, introduces less computation overhead due to the combination calculations. In real group linkage cases, it is also reasonable to select a large similarity threshold since there is no need to link two groups that are not similar.

Then, we use the same group settings to evaluate the performance of the TPPGLE protocol using homomorphic encryption (TPPGLE-H). We adopt the Paillier encryption scheme to encrypt the coefficients used in Protocol 4. The average end-to-end execution time under different group sizes are shown in Figures 3(e)-(g). The results show that the performance does not change much under different thresholds, but is greatly affected by different group sizes.

Therefore, we run another experiment to evaluate the efficiency of two TPPGLE protocols over larger groups. We set the threshold to $\theta = 0.7$, and generate 100 pairs of groups with different sizes (5, 10, 15, 20, and 25). Figure 4(a) shows the average end-to-end execution time for both commutative encryption based approach and homomorphic encryption based approach. Apparently, when the group size increases, computation cost of the TPPGLE-H protocol shows a linear increasing trend, while in the TPPGLE-C protocol it grows almost exponentially. This is because in TPPGLE-H protocol, the computational complexity increases along with the degree of the group's polynomial representation, and thus increases linearly with the group size. In TPPGLE-C protocol, its computational complexity depends on the k-combination function: $Cost(n) = C_k^n \times \mathbf{E}(*)$. For a given k, the computation cost increases with n in a manner slower than exponential but faster than polynomial.

7.2 TPPGLA

We evaluate the validity and efficiency of TPPGLA protocol (Protocol 6) on the movie network data set. To form the group, we randomly extract a subset of

1000 records from MN, and calculate pairwise record-level similarities between the records. For a given record-level cut-off ρ (a preset value negotiated between Alice and Bob), we divide the records into two parts, according to whether record pairs are similar or not.

We first select k pairs of different records from the similar set as input, where k is the set-intersection threshold, and apply Protocol 6 on the k pairs. The output is "Yes", which verify the validity of the protocol. Then, we select random group pairs from MN to evaluate the efficiency of the protocol, and show the computation cost of TPPGLA under different group sizes (from 3 to 11) in Figure 4(b). From the results, we see that the computation cost increases greatly with the group size. This is because the computation cost is proportionate to group size $m \times n$ and the dimensionality of vector space, while the latter also increases with the group size. Overall, TPPGLA introduces a comparably large overhead in end-to-end execution time , however, it is the price we pay for extreme cases with strong needs for privacy protection.

8 Conclusion and Future Works

In this paper, we have presented privacy preserving group linkage, in which groups of records from two parties are compared in a way that no record content is revealed. Simple PPGL (in which both parties learn the group similarity) suffers from the group membership inference problem, which could be employed to learn the member records of the other party's groups, even though the groups are not linked. To tackle the problem, we propose threshold privacy preserving group linkage, in which both parties only learn the verdict on whether the two groups are matched or not, instead of the value of group similarity. We implemented and tested TPPGL protocols for both exact matching and approximate matching scenarios. From the experiments, we can see that TPPGL pays a price in computation in order to protect the participants' privacy.

Although our approach demonstrates strong privacy protection, the computation overhead is relatively high. In its current form, the approaches are suitable for exchanging highly sensitive information. Our future work is to further explore cryptographic methods to reduce the overall computation of our approaches. Meanwhile, we are also optimizing our existing implementations, and planning to test it over large datasets.

References

1. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: SIGMOD (2003)
2. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)

3. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE TKDE, 19, 1–16 (2007)
4. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
5. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
6. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178. ACM, New York (2009)
7. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
8. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: STOC 1985, pp. 291–304 (1985)
9. Guha, S.: Merging the results of approximate match operations. In: VLDB, pp. 636–647 (2004)
10. hai Do, H., Rahm, E.: Coma - a system for flexible combination of schema matching approaches. In: VLDB, pp. 610–621 (2002)
11. Hall, R., Fienberg, S.E.: Privacy-preserving record linkage. In: Domingo-Ferrer, J., Magkos, E. (eds.) PSD 2010. LNCS, vol. 6344, pp. 269–283. Springer, Heidelberg (2010)
12. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: SIGMOD (1995)
13. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: ICDE, pp. 496–505 (2008)
14. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: EDBT, pp. 123–134 (2010)
15. Jaccard, P.: Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Société Vaudoise des Sciences Naturelles 37, 547–579 (1901)
16. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
17. Murugesan, M., Jiang, W., Clifton, C., Si, L., Vaidya, J.: Efficient privacy-preserving similar document detection. The VLDB Journal 19, 457–475 (2010) 10.1007/s00778-009-0175-9
18. Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: CCS, pp. 59–66. ACM, New York (1998)
19. On, B.-W., Koudas, N., Lee, D., Srivastava, D.: Group linkage. In: IEEE ICDE, Istanbul, Turkey, pp. 496–505 (April 2007)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Paillier, P., Pointcheval, D.: Efficient public-key cryptosystems provably secure against active adversaries. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 165–179. Springer, Heidelberg (1999)
22. Scannapieco, M., Figotin, I., Bertino, E., Elmagarmid, A.K.: Privacy preserving schema and data matching. In: SIGMOD, pp. 653–664. ACM, New York (2007)

23. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: KDD, pp. 807–816 (2009)
24. Tejada, S., Knoblock, C.A.: Learning domain-independent string transformation weights for high accuracy object identification. In: ACM SIGKDD, pp. 350–359 (2002)
25. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
26. Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. SIGMOD Rec. 33(1), 50–57 (2004)
27. Winkler, W.E.: Overview of record linkage and current research directions. Technical report, Bureau of the Census (2006)
28. Yao, A.C.: Protocols for secure computations. In: SFCS 1982: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE Computer Society, Washington, DC, USA (1982)