Efficient Private Record Linkage§

Mohamed Yakout, Mikhail J. Atallah, Ahmed Elmagarmid

Department of Computer Sciences, Purdue University
West Lafayette, IN 47907, USA
{myakout,mja,ake}@cs.purdue.edu

***Abstract*— Record linkage is the computation of the associations among records of multiple databases. It arises in contexts like the integration of such databases, online interactions and negotiations, and many others. The autonomous entities who wish to carry out the record matching computation are often reluctant to fully share their data. In such a framework where the entities are unwilling to share data with each other, the problem of carrying out the linkage computation without full data exchange has been called *private record linkage*. Previous private record linkage techniques have made use of a third party. We provide efficient techniques for private record linkage that improve on previous work in that (*i*) they make no use of a third party; (*ii*) they achieve much better performance than that of previous schemes in terms of execution time and quality of output (i.e., practically without false negatives and minimal false positives). Our software implementation provides experimental validation of our approach and the above claims.**

I. INTRODUCTION

Record linkage is the process of identifying similar records that represent the same real world entity. The problem of matching records among sources that are autonomous and unwilling to share data is known as *private record linkage* [3].

Approximate record matching is difficult, especially when privacy is an additional constraint. There are mainly three approaches adopted for solving this problem. First, the use of cryptographic methods [2] [6] [7]: while it guarantees accurate results with high privacy, it is not practical to be used for large databases and for approximate matching. Second, applying perturbation methods on private information to obscure individual identity [9]: while this approach is cost efficient, it lacks the required accuracy for linking records. Finally, non-cryptographic techniques using an outside third-party [4] [5]: it does not require that either side knows the data from the opposing side. However, a third party facilitates the task of linkage without colluding with either party against the other. It has traditionally been assumed that the participants are "honest but curious" in the sense that they do want to compute a correct answer and will follow the protocol's steps. However, while they do not want to reveal their own data to the other party, they can try to figure out the other party's data from their transcript of the protocol's execution (so it is up to the protocol to prevent this).

Our protocol consists of two phases. The main goal of Phase 1 is to produce candidate pairs of records for matching, by carrying out a very fast but not accurate matching between pairs of records. The technique used in Phase 1 is of independent interest, and includes a novel "winnowing" technique for finding a small set of candidate pairs with few false positives and no false negatives. Phase 2 uses a practical privacy-preserving protocol for completing the task of computing the Euclidean distance between each candidate pair; both parties participate in its computations without revealing the original vector representations of their respective records.

The remainder of the paper is organized as follows: A formal definition for the problem is presented in section II. In section III, we describe the protocol steps and the algorithms used. Afterward, we present the experiments conducted to study and evaluate protocol efficiency in section V.

II. PROBLEM FORMULATION

Record linkage is a process of identifying record pairs across two databases that both correspond to the same real-world entity. The process includes building a *Matching Function* that takes as input a set of thresholds, and pair of records to classify as *match* or *mismatch* according to a predefined decision rule.

Definition: Matching function

Given two relations with the same attributes R_A (a_1, a_2 ..., a_t) and R_B (a_1, a_2 ..., a_t). A matching function *MF* takes as input triple $(r_A, r_B, \{\theta_1,...\theta_t\})$ and produces a Boolean output {*True*, *False*} corresponding to {*match*, *mismatch*}, where:

- $r_A \in R_A$ is a record with attribute values $(r_A(a_1),...,r_A(a_t))$ and $r_A(a_1) \in Dom(R_A.a_1)$, ..., $r_A(a_t) \in Dom(R_A.a_t)$.
- $r_B \in R_B$ is a record with attribute values $(r_B(a_1),...,r_B(a_t))$ and $r_B(a_1) \in Dom(R_B.a_1),...,r_B(a_t) \in Dom(R_B.a_t)$.
- $\{\theta_1,...\theta_t\}$ are predefined similarity thresholds for the corresponding attributes a_1, ... a_t in R_A and R_B.

The output of the matching function *MF* is decided based on

$$MF(r_A, r_B, \{\theta_1,...\theta_t\}) = \begin{cases} True & iff \quad \bigwedge_{i=1}^{t} f_i(r_A(a_i), r_B(a_i)) \le \theta_i \\ False & otherwise \end{cases}$$

where $f_i : Dom(R_A.a_i) \times Dom(R_B.a_i) \rightarrow \Re^+$, $i = 1,...,t$, are predefined similarity measures or distance functions defined over the domains of corresponding attribute a_i for the relations R_A and R_B (refer to [8, 1] for surveys of distance functions).

The problem addressed in this paper can be defined as follows: Let *A* and *B* be two parties owning the relations R_A and R_B respectively. The private record linkage is to build a matching function *MF* that identifies the matched records between R_A and R_B and operates in a privacy preserving manner, such that at the end of the process *A* and *B* will know

§ This work was supported by grants from NSF-ITR 0428168, NSF-CNS 0627488, the Lilly Endowment, US DHS (PURVAC), and by the sponsors of CERIAS.

only a set of matched records in R_A and R_B respectively and no information will be revealed about the mismatched records.

III. PROTOCOL

The overall process of the protocol is illustrated in Fig. 2, to which we will refer as we cover its main stages.

Step 1 in Fig. 2 depicts the transformation of the database records into vectors. This is done using the described protocol in [5]. We henceforth assume that this transformation to numeric vectors has already been done by each party; so our algorithms and protocols deal with numeric vectors.

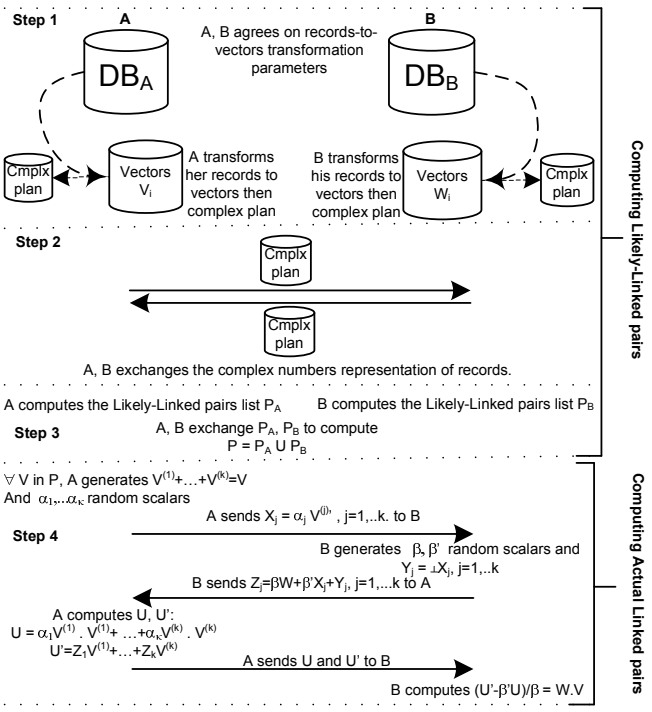


Fig. 1 Protocol overall process.

A. Computing Likely Linked Pairs

This section describes our scheme for computing pairs of vectors that are likely to be linked, practically without any false negatives and with minimal false positives. In what follows uppercase symbols are vectors, lowercase are scalars.

The m vectors with party A are denoted as $V_1, \ldots, V_m$, and the n vectors with party B are denoted as $W_1, \ldots, W_n$. We assume $V_i = (v_{i,1}, \ldots, v_{i,d})$ and $W_i = (w_{i,1}, \ldots, w_{i,d})$.

The algorithm consists of a first stage that maps each of the vectors to a point in the complex plane, and the second stage computes the close pairs in the complex plane.

Algorithm Likely_Linked

Inputs: A has d-dimensional vectors $V_1, \ldots, Vm$, B has d-dimensional vectors $W_1, \ldots, W_n$.

Output: A set P of pairs of indices i, j such that V_i and W_j are likely to be linked. P includes all vector pairs to be actually linked, and some pairs that will not be linked.

Algorithm steps:

1- For every vector V_i, A computes a complex number c(V_i) obtained as follows:

$$c(V_i) = \sum_{k=0}^{d-1} v_{i,k} e^{2k\pi\sqrt{-1}/d}$$

B similarly computes c(W_j) for every W_j:

$$c(W_j) = \sum_{k=0}^{d-1} w_{j,k} e^{2k\pi\sqrt{-1}/d}$$

2- A and B compute the set P of all pairs i, j such that c(V_i), c(W_j) are closer than a proximity threshold δ; δ is chosen based on both the data and on what is to be considered "close" (a coarse notion of closeness implies a larger choice of δ and a larger number of pairs in P; the experimental work section discusses our choice of δ). There are two ways of computing P: (i) The parties consider a c(X) to hide enough of the private original record (from which X is derived) that they do not mind sharing c(X) with each other; and (ii) the parties are unwilling to exchange their respective c(X)'s. We assume (i) in what follows because it is the most likely practical scenario (we can accommodate using (ii) but it would require using a 2-dimensional version of the distance protocol we give in the next section, within a cumbersome modification of the algorithm below). Hence we assume, in the following steps of the algorithm for computing P, that A and B have already exchanged their c(V_i)'s and c(W_i)'s (this is denoted as step 2 in Fig.2). We use $Re(p)$ to denote the real ("horizontal") component of a complex number p, $Im(p)$ to denote its imaginary ("vertical") component. We use p_i as a shorthand for c(V_i), and q_i as a shorthand for c(W_j).

a) A (B) initializes P_A (P_B) to be empty. Note. P_A (P_B) will eventually contain all pairs i, j such that p_i (q_i) is to the left of q_j (p_j) and the distance between them is $\leq \delta$.

b) A (B) sweeps a vertical slab S_A (S_B) of width δ in the left-to-right direction (from $-\infty$ along the real axis to $+\infty$) and maintains, during the sweep, all the p_is (q_is) that lie in S_A (S_B) in a dynamic tree data structure T_A (T_B) according to their $Im(p_i)$ ($Im(q_i)$) values (see Fig. 3). T_A (T_B) starts by being empty, when S_A (S_B) starts out at $-\infty$ along the real axis. They evolve and are queried as described in the next step.

c) As S_A (S_B) moves left to right, an event is when a p_i (q_i) enters or exits S_A (S_B), or when a q_i (p_i) enters S_A (S_B). These three types of events are handled differently: (i) When the leading vertical boundary of the sweeping S_A (S_B) touches a p_i (q_i), that p_i (q_i) gets inserted in T_A (T_B) because it is entering S_A (S_B); (ii) when the trailing vertical boundary of the sweeping S_A (S_B) touches a p_i (q_i), that p_i (q_i) gets deleted from T_A (T_B) because it is exiting S_A (S_B); (iii) When the leading vertical boundary of the sweeping S_A (S_B) touches a q_j

(p_j), A (B) locates $Im(q_j)$ ($\mathrm{Im}(p_j)$) in T_A (T_B), computes the distance between p_i (q_i) and every q_j (p_j) whose imaginary component is within of $Im(q_j)$ ($Im(p_j)$), and includes every pair i, j for which that distance is $\leq \delta$ in P_A (P_B).

d) A and B exchange their P_A and P_B, then each computes $P = P_A \cup P_B$.

Correctness: We now sketch correctness of the above algorithm. It suffices to prove that P_A (P_B) contains all the pairs i, j for which p_i (q_i) is to the left of q_j (p_j). Suppose, w.l.o.g, that p_i is to the left of q_j , and that they are no more than a distance of δ apart. When S_A encounters p_i, p_i gets inserted into T_A. When, later on, S_A encounters q_j , we have that :

- p_i is still in T_A because otherwise it would be farther than δ away from q_j ; and
- $Im(p_i) \in [Im(q_j) - \delta, Im(q_j) + \delta]$ because otherwise p_i and q_j would be farther than δ apart. This means that the query on T_A in the above Sub-step 2(c) (using $Im(q_j)$ as search key) does return p_i.

The above implies that the pair i, j gets put into P_A. If p_i had been to the right of q_j then a similar argument shows that i, j gets put into P_B. Therefore, every p_i and q_j that are within δ from each other results in a pair i, j being put in P_A, or in a pair i, j being put in P_B. Therefore, all pairs within δ of each other in the complex plane are found.

Efficiency: The overall time complexity is easily seen to be $O(t + m \log m + n \log n)$ where t is the number of pairs whose distance in the complex plane gets computed, which is close to the number (call it t') of pairs that end up in P, which is the number of pairs i, j for which the L_2 distance between p_i and q_j is $\leq \delta$. Assuming a uniform distribution for the p_is and q_is, it is straightforward to show that t is only a multiplicative factor of $4/\pi$ away from the number t', and therefore it is $O(t')$.

B. Computing Actually Linked Pairs

Step 4 in Fig. 2 computes the distance between V_i and W_j for every pair i, j of P produced by the algorithm of the previous section, and decides whether V_i and W_j are linked based on the computed distance. To compute this distance privately and without using expensive cryptographic operations (like public key encryptions) we present the following protocol.

Scalar Product Protocol

Inputs: A has vector V , B has vector W. The vectors are d-dimensional where d > 2. Both have an integer security parameter k.

Output: B learns $W \cdot V$.

Protocol steps:

1- A generates k-1 linearly independent random vectors $V^{(1)}, \ldots, V^{(k-1)}$, computes $V^{(k)} = V - V^{(1)} \ldots - V^{(k-1)}$.

2- For $j = 1, \ldots, k$, A computes a random α_j and creates vector $X_j = \alpha_j V^{(j)}$. She sends $X_1, \ldots, X_k$ to B.

3- B generates random scalars β and β', and k random vectors $Y_1, \ldots, Y_k$ where Y_j is orthogonal to X_j, $j = 1, \ldots, k$. B sends to A the k vectors

$$Z_j = \beta W + \beta' X_j + Y_j, \quad j = 1, \ldots, k.$$

4- A computes and sends to B the scalars U and U':

$$U = \alpha_1 V^{(1)} \cdot V^{(1)} + \ldots + \alpha_k V^{(k)} \cdot V^{(k)}$$

$$U' = Z_1 \cdot V^{(1)} + Z_2 \cdot V^{(2)} + \ldots + Z_k \cdot V^{(k)}$$

$$= \beta W \cdot V + \beta' \alpha_1 V^{(1)} \cdot V^{(1)} + \ldots + \beta' \alpha_k V^{(k)} \cdot V^{(k)}$$

where we simplified using the fact that $Y_j \perp V^{(j)}$.

5- B computes $(U' - \beta' U)/\beta$, which equals $W \cdot V$.

Note that B learns only (i) a k-dimensional hyperplane that contains V and that is selected (in Step 1) by A; and (ii) the scalar U (in Step 4). For (i), using a larger k makes it possible for A to increase privacy. Moreover, for a given k, A can choose a relatively innocuous hyperplane to reveal (not all components of a vector are equally private, and our protocol makes it possible to mask the more private ones at the expense of the less private ones). For (ii), the fact that all of the α_js are unknown to B provides the privacy.

In step 3, A knows X_j but does not know the scalars β, β' or the vectors Y_j, which hide W from A (because B is effectively adding a random vector of B's choice to W for hiding it). Note that without the $\beta' X_j$ then, A could obtain the direction of vector W in space (but not its magnitude) by computing the k scalar products $Z_j.V^{(j)} = \beta\, W.V^{(j)}$ (their ratios would reveal that direction, as β cancel out).

IV. EXPERIMENTS

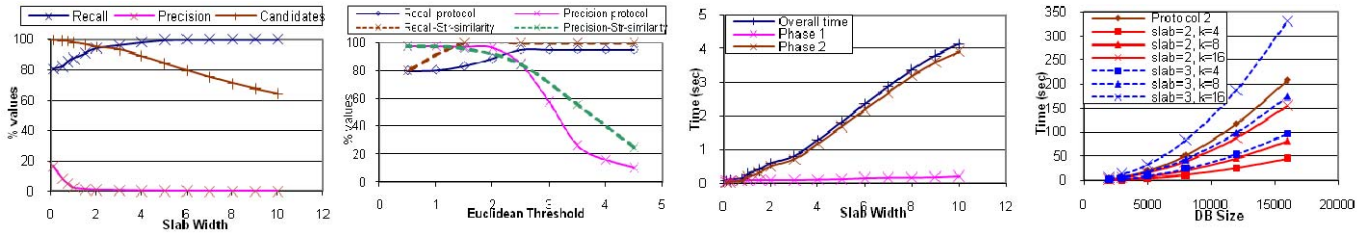
We performed experiments with the following goals:

- Illustrating the accuracy of the data matching. We analyzed the effect of the likely-linked pairs' computation on data matching in terms of the classical precision and recall metrics.
- Evaluating the performance speed of the protocol and studying the effect of the slab width used in computing the likely-linked pairs. Moreover, the protocol performance is compared with the one described in [5].

We used a real-world dataset represents a British Columbia voters' list containing 34,261 records of voters' names and addresses (http://www.rootsweb.com/~canbc/vote1898). We used only the first and last name fields. The dataset has been partitioned into several pairs of datasets with different sizes to be linked. Also, we managed to identify and control the percentage of similar records between each dataset pair.

A. Accuracy Study

The matching accuracy of the proposed protocol is studied by reporting precision and recall of matched pairs. In this experiment, we link two datasets of size about 1000. In Fig. 2.a, we illustrate the effectiveness of the likely-linked phase. We report the recall, precision and the percentage of reduction in the number of generated candidates against the slab width.



(a) Phase 1 analysis: Dataset 1 (b) Overall accuracy: Dataset 1 (c) Execution time analysis. (d) Exe. time comparison with Protocol2

Fig. 2 Experiments.

It is noted that high recall values are maintained especially for large slab width and slightly decreases for small slab widths less than 3. This is expected since in the likely-linked phase, the records representation is not accurate and hence, very small slab widths allow for having more false negatives matches. On the other hand, the precision takes very low values. When using small slab width only very similar and exact matched pairs are detected. For slab width between 2 and 3, note the effectiveness in reducing the number of candidates to around 90% for the accurate matching phase, while maintaining high recall above 98%.

The overall protocol's accuracy is depicted in Fig. 2.b. We used a slab of size 3 and passed the candidates to the accurate matching phase. We reported the overall protocol's recall and precision against the Euclidean distance similarity threshold. For threshold between 2 and 2.5, a significant improvement in the precision is illustrated while maintaining high recall. Consequently, the loss in the matching accuracy due to the records transformation and running our protocol is acceptable.

To conclude, the slab width is the parameter that helps in providing a trade-off between the protocol accuracy and cost. Increasing the slab width guarantees high recall while producing a lot of candidates for Phase 2 and consequently, slow performance, and vice versa.

B. *Performance Study*

For the performance, we first study the performance sensitivity to the slab width used in the protocol. Second, we compare our protocol performance with the one described in [5], referred to as Protocol 2, with the advantage that our protocol does not require a third party, while not sacrificing the privacy.

Fig. 2.c shows the protocol execution for different values of slab width. In this experiment, we report the overall time in addition to the time taken by each of the protocol's two phases. It is noted that the execution time increases linearly with the increase in the slab width. This is due to the increase in the number of candidates for Phase 2. It is also noted that likely-linked phase execution time is very small. Increasing the slab width implies an increase in the number of likely-linked pairs and, consequently, an increase in the overall execution time.

We have compared the execution time of our protocol performance with Protocol 2. Protocol 2 performs the private record linkage. It uses the same records to vectors transformation we use, and then the vectors are shipped to a third party to perform the actual records matching based on the Euclidean distance between the vectors.

In Fig. 2.d, we compare the execution time of our protocol to Protocol 2. Several dataset pairs were used in this experiment and we report the execution time using slab of widths 2 and 3. These slab widths were selected for better accuracy as illustrated in Fig 2.a. We also used three different values 4, 8 and 16 for *k* (tunable privacy parameter). Using the two slab widths, our protocol achieves better performance than Protocol 2 except when slab = 3 and k = 16. This is due to the increase in the number of candidates in addition to more computation for scalar products. Protocol 2 performs *n* x *m* record pairs matching using the expensive Euclidean distance evaluation. On the other hand, our protocol uses Phase 1 to quickly reduce the number of required pairs to be matched and divide the efforts among the participants. Moreover, despite the fact that our protocol is based upon computing the same Euclidean distance in a privacy preserving manner, our protocol's total execution time is smaller.

V. CONCLUSION

We gave a protocol for private record linkage that outperforms previous work in two ways: (i) it eliminates the need for a third party; and (ii) it is much faster. The techniques we use in achieving the result are of independent interest, in particular our private distance computation protocol that is the first to be fast enough for use in applications involving massive data.

REFERENCES

[1] A.K. Elmagarmid, G.I. Panagiotis, S.V. Verykios, "Duplicate Record Detection: A survey", *IEEE TKDE* 19 (2007), no. 1.

[2] R. Agrawal, A. Evfimievski, R.Srikant, "Information Sharing Across Private Databases", In *Proc. SIGMOD*, 2003.

[3] W. Du and M. Atallah. "Potocols for secure remote database access with approximate matching". In *1st Workshop on Security and Privacy in E-Commerce*, 2000.

[4] A. Al-Lawati, D. Lee, P. McDaniel, "Blocking-aware Private Record Linkage", In *Proc. IQIS*, 2005.

[5] M. Scannapieco, I. Figotin, E. Bertino, A. K. Elmagarmid, "Privacy preserving schema and data matching". *SIGMOD Conference*, 2007, 653-664.

[6] M.J. Freedman, K. Nissim, B. Pinkas, "Efficient Private Matching and Set Intersection", In *Proc. EUROCRYPT* , 2004.

[7] L. Kissner, D.Song, "Private and Threshold Set-intersection", Tech. Report CMU-CS-05-113, 2005.

[8] N. Koudas, S. Sarawagi, D. Srivastava, "Record Linkage: Similarity Measures and Algorithms", In *Proc. SIGMOD*, 2006.

[9] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, pp. 557-570, 2002.