Privacy Preserving
Schema and Data Matching

Monica Scannapieco
ISTAT and SAPIENZA – Univ. Roma, ITALY
scannapi@istat.it

Ilya Figotin
Purdue University, USA
ifigotin@cs.purdue.edu

Elisa Bertino
Purdue University, USA
bertino@cs.purdue.edu

Ahmed Elmagarmid
Purdue University, USA
ake@cs.purdue.edu

ABSTRACT

In many business scenarios, record matching is performed across different data sources with the aim of identifying common information shared among these sources. However such need is often in contrast with privacy requirements concerning the data stored by the sources. In this paper, we propose a protocol for record matching that preserves privacy both at the data level and at the schema level. Specifically, if two sources need to identify their common data, by running the protocol they can compute the matching of their datasets without sharing their data in clear and only sharing the result of the matching.The protocol uses a third party, and maps records into a vector space in order to preserve their privacy.Experimental results show the efficiency of the matching protocol in terms of precision and recall as well as the good computational performance.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; H.2.5 [**Database Management**]: Heterogeneous Databases

General Terms

Security

Keywords

Record matching, Privacy

1. INTRODUCTION

Record matching is the process of identifying if two (or more) records represent the same real world entity or not. Within a single source, record matching is executed for identifying and eliminating duplicate records. Across sources, record matching can be performed for improving the data quality of a source by means of correction by comparisons with a better quality source, and also for the purpose of integrating data.

In this paper, we focus on record matching performed with the aim of identifying common information shared by two data sources. In this case, records referring to the same entity can represent that entity differently due to any of the following factors: different keys (key-conflicts), errors about the values of the record attributes (attribute-conflicts) and possibly schema-level inconsistencies, such as different descriptions (formats, types, etc.), different models, and different structural representations. Several applications may require distinct data sources to exchange and share common data. For instance consider an eGovernment scenario, in which an application wants to make controls about individuals that are suspected of tax evasion. The application may require the tax agency to share data concerning suspected individuals with other databases, e.g. a customer database of a company selling luxury goods; a record matching between the agency's database and the company's one can link the same individuals and help the tax control process. Though on one side such applications need to share and integrate data, on the other side they have to deal with a major and legitimate issue: protecting the privacy of the individuals to whom such data is related. In the above example, the tax agency cannot simply send data related to the suspected individuals, but it must adopt techniques for preserving their privacy.

In this paper, we propose a privacy-preserving protocol to perform record matching across two data sources, which is more efficient than protocols based on cryptographic techniques such as privacy-preserving set intersection. Our protocol has an interesting new feature that concerns the privacy preservation of database schemas. Whereas the problem of privacy protection of data has been investigated, much less attention has been devoted to schema level information. If two companies decide to match data concerning their respective customers, they first have to agree on a common schema format to use for the actual *data* matching. However, this means that they must reveal to each other how they store their own customer information. In many cases, companies may not be willing to give away a competitive advantage by revealing such information.

Our protocol performs record matching by using an approach that preserves privacy of both data and schema and has two major innovations with respect to previous approaches. With respect to approaches dealing with private data sharing, such as the one proposed by Agrawal et al.

SIGMOD'07, June 12–14, 2007, Beijing, China.
Copyright 2007 ACM 978-1-59593-686-8/07/0006 ...$5.00.

[2], that only perform exact matching, our protocol performs privacy-preserving *approximate matching*. It is important to notice that when data across different sources have heterogeneous quality, an exact match may not be very successful, as it may likely result in very few matches. In such cases, approximate match is the only viable approach. Approximate matching is however an indirect, somewhat complicated process, as it requires the computation of distance functions among records, the values of which have to be kept private. As an example, computing a distance `dist(`a_1`,`a_2`)`, where a_1 and a_2 are two data items owned respectively by sources P and Q, requires both values to be available at the same time to one party. However, under privacy constraints, such simple condition cannot be met in that P cannot see a_2 and Q cannot see a_1. Even if a third neutral party is introduced, such party can compute neither the distance between the plain values of a_1 and a_2, nor the distance between encrypted values, because encryption functions do not generally preserve similarity distances. The second major innovation is related to the *awareness of schema information*. Beyond preserving schema privacy, we exploit schema information throughout the instance matching process. This is a major enhancement provided by our protocol when compared to approaches based on secure set intersection, such as the approach by Naor et al. [19]. These approaches typically consider only sets of simple objects, such as strings or points. Therefore the comparison among these objects is typically straightforward. By contrast our approach considers sets of records that have: *(i)* an intensional structure that deeply affects the instance matching protocol and *(ii)* schema level heterogeneities. Another important characteristic of our approach is that, unlike those protocols, it is efficient also for large databases. For instance the secure set intersection protocol by Naor et al. [19] has a complexity in $O(n^2)$, where n is the database size, thus making this protocol not applicable to databases. Furthermore, our technique does not rely on the usage of complex cryptographic processes, but assures privacy by application of a method widely used for similarity based searching of complex objects. The main idea of our work is to embed records to be matched in an Euclidean space, that is, a vector space having the Euclidean distance as norm, and to perform the comparison in such a space. The embedding method we use assures privacy in that it is based on a random selections of the axes space. The usage of a metric space is typically made when comparing images, documents, and complex objects for which performing comparison in the original space is time consuming. Jin et al. [15] have used such an approach applied to records rather than to complex objects and shown its efficiency and flexibility with respect to the use of several similarity functions. However, because this proposal does not preserve privacy, we adopt SparseMap [14], a different embedding method which provides privacy guarantees and has important features distinguishing it from FastMap [14], the embedding method used by Jin et al. [15].

The rest of the paper is organized as follows. Section 2 defines the addressed problem. Sections 3 and 4 provide the detail of the instance and schema matching protocols, which are combined in a single one in Section 5. Section 6 analyzes the security of the protocol and Section 7 shows its experimental evaluation. The discussion of related work (Section 8) and future work (Section 9) concludes the paper.

2. PROBLEM FORMULATION

In this section, we define the problem investigated in the paper. We start by defining the notion of comparison function and matching decision rule, on the basis of which we define record matching and the problem of private record matching addressed in the paper.

Definition - Comparison Function. *Let D be a domain. A comparison function over D is defined as $f : Dom(D) \times Dom(D) \rightarrow \mathbb{R}^+$. f takes as input two values from domain D and returns a positive real value corresponding to the* distance *between the input values.*

The definition of comparison function can be detailed by defining the notion of distance which is used; examples of proposed distances include edit distance, Smith-Waterman distance and Jaro distance (see [17] for a survey). As an example, the edit distance between two strings is defined as the minimum cost to convert one string into another by a sequence of character insertions, deletions, and replacements. Assuming that the insertion cost and the deletion cost are each equal to 1, the edit distance between the strings `Smith` and `Sitch` is 2, as `Smith` is obtained by adding `m` and deleting `c` from `Sitch`. Comparison functions are used to compare values of attributes of specific records, in order to decide if they match or not. Matching decision rules have the purpose of classifying pairs of records as a match or a non-match on the basis of the similarities computed on each corresponding pair of field values. Decision rules can be quite complex and be extracted from domain knowledge. In the following definition, we consider a matching decision rule, which essentially checks the logical `AND` of similarity threshold-based conditions on pairs of record attribute values.

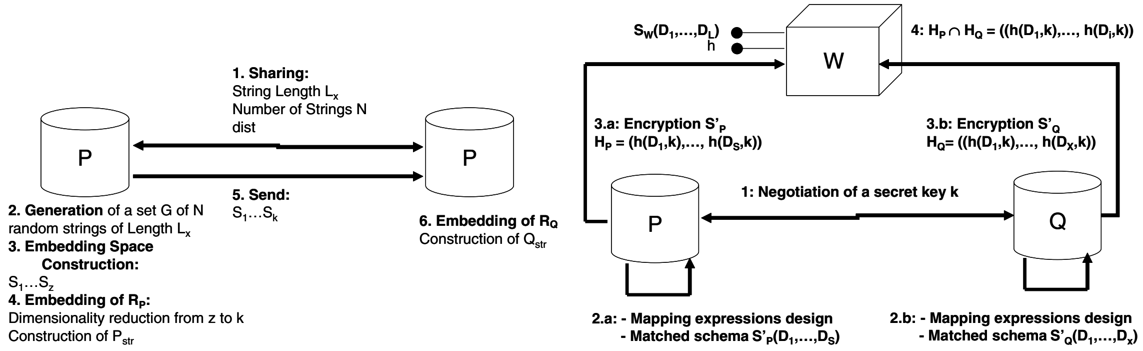
Definition - Matching Decision Rule. *Given two relations $R(A_1, \ldots, A_n)$ and $S(A_1, \ldots, A_n)$, a set of n similarity functions f_i and a set of n positive real numbers θ_i called* similarity thresholds*, a matching decision rule $DR_{R,S}$ is a function $DR_{R,S} : R \times S \rightarrow \{TRUE, FALSE\}$ such that, for each record $r(a_1, \ldots, a_n) \in R$ and $s(a_1, \ldots, a_n) \in S$*

$$DR_{R,S}(r,s) \;=\; \begin{cases} TRUE & iff\ (f_1(r.a_1, s.a_1) \leq \theta_1) \bigwedge \\ & \ldots \bigwedge (f_n(r.a_n, s.a_n) \leq \theta_n) \\ FALSE & otherwise \end{cases}$$

If $DR_{R,S}(r,s) = TRUE$, then the pair (r,s) is a MATCH, otherwise it is a NONMATCH.

Definition - Record Matching. *Given two relations $R(A_1, \ldots, A_n)$ and $S(A_1, \ldots, A_n)$, a record matching process compares records in R with records in S on the basis of a matching decision rule $DR_{R,S}$, and determines:* (i) *the set of matched records, including all the pairs of R's records and S's records for which $DR_{R,S}(r,s) = TRUE$ and* (ii) *the set of unmatched records, including all the pairs of R's records and S's records for which $DR_{R,S}(r,s) = FALSE$.*

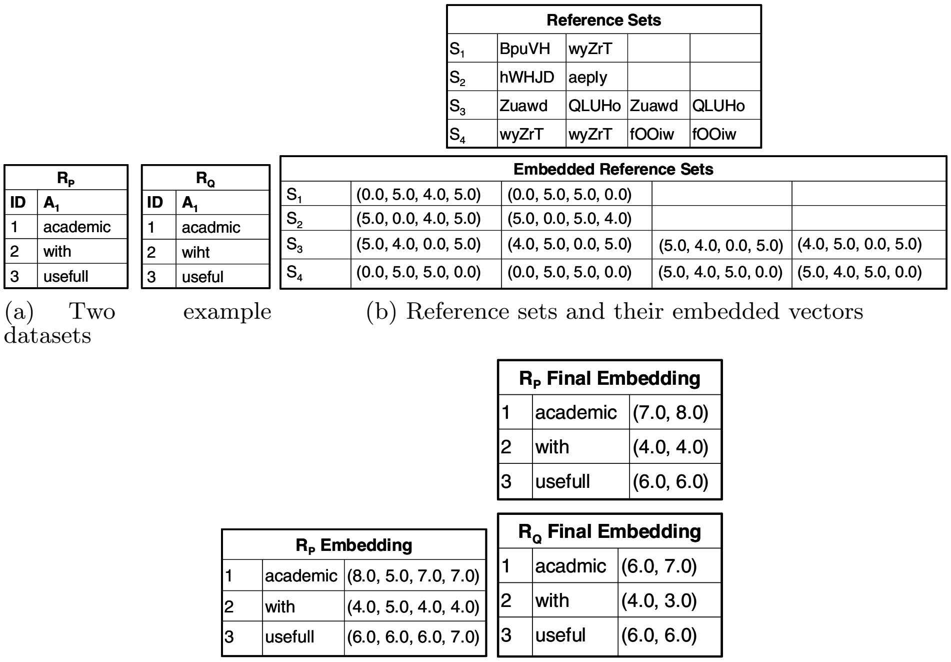
Note that we assume that no duplicates are present in any of the two relations R and S; indeed, a de-duplication activity can be reasonably executed by each source before starting the record matching process with the other source. We now formulate the problem of private record matching. Let P and Q be two parties. Consider the two following hypotheses.


1. Sharing:
String Length L_x
Number of Strings N
dist
P
P
2. Generation of a set G of N random strings of Length L_x
3. Embedding Space Construction:
$S_1 \ldots S_z$
4. Embedding of R_P:
Dimensionality reduction from z to k
Construction of P_{str}
5. Send:
$S_1 \ldots S_k$
6. Embedding of R_Q
Construction of Q_{str}
$S_W(D_1,\ldots,D_L)$
h
W
4: $H_P \cap H_Q = ((h(D_1,k),\ldots, h(D_i,k))$
3.a: Encryption S'_P
$H_P = (h(D_1,k),\ldots, h(D_S,k))$
3.b: Encryption S'_Q
$H_Q= ((h(D_1,k),\ldots, h(D_X,k))$
1: Negotiation of a secret key k
P
Q
2.a: - Mapping expressions design
- Matched schema $S'_P(D_1,\ldots,D_S)$
2.b: - Mapping expressions design
- Matched schema $S'_Q(D_1,\ldots,D_x)$

(a) Phase 1 and 2 of the private data matching protocol

(b) Steps of the schema matching protocol

Figure 1: Data and schema matching protocols



Reference Sets				
S_1	BpuVH	wyZrT		
S_2	hWHJD	aeply		
S_3	Zuawd	QLUHo	Zuawd	QLUHo
S_4	wyZrT	wyZrT	fOOiw	fOOiw

R_P	
ID	A_1
1	academic
2	with
3	usefull

R_Q	
ID	A_1
1	acadmic
2	wiht
3	useful

Embedded Reference Sets				
S_1	(0.0, 5.0, 4.0, 5.0)	(0.0, 5.0, 5.0, 0.0)		
S_2	(5.0, 0.0, 4.0, 5.0)	(5.0, 0.0, 5.0, 4.0)		
S_3	(5.0, 4.0, 0.0, 5.0)	(4.0, 5.0, 0.0, 5.0)	(5.0, 4.0, 0.0, 5.0)	(4.0, 5.0, 0.0, 5.0)
S_4	(0.0, 5.0, 5.0, 0.0)	(0.0, 5.0, 5.0, 0.0)	(5.0, 4.0, 5.0, 0.0)	(5.0, 4.0, 5.0, 0.0)

(a) Two example datasets

(b) Reference sets and their embedded vectors

R_P Final Embedding		
1	academic	(7.0, 8.0)
2	with	(4.0, 4.0)
3	usefull	(6.0, 6.0)

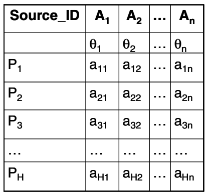
R_P Embedding		
1	academic	(8.0, 5.0, 7.0, 7.0)
2	with	(4.0, 5.0, 4.0, 4.0)
3	usefull	(6.0, 6.0, 6.0, 7.0)

R_Q Final Embedding		
1	acadmic	(6.0, 7.0)
2	with	(4.0, 3.0)
3	useful	(6.0, 6.0)

(c) R_P embedding

(d) R_P and R_Q final embeddings

Figure 2: Application of phases 1 and 2 to an example dataset



Source_ID	A_1	A_2	...	A_n
	θ_1	θ_2	...	θ_n
P_1	a_{11}	a_{12}	...	a_{1n}
P_2	a_{21}	a_{22}	...	a_{2n}
P_3	a_{31}	a_{32}	...	a_{3n}
...	...	...	...	...
P_H	a_{H1}	a_{H2}	...	a_{Hn}

Figure 3: Elements in the P_{str} structure

Hypothesis 1. Parties P and Q store the records to be matched in the two relations $R_P(A_1, \ldots, A_n)$ and $R_Q(B_1, \ldots, B_n)$ respectively, having *identical* schemas.

Hypothesis 2. Parties P and Q store the records to be matched with possible schema-level integration conflicts [22].

The problem addressed in this paper can be formulated as follows.

Problem Statement. Let P and Q be two parties owning relations $R_P(A_1, \ldots, A_n)$ and $R_Q(B_1, \ldots, B_n)$, respectively. The *privacy-preserving record matching* problem is to perform record matching between R_P and R_Q, such that at the end of the process P will know only a set P_{Match}, consisting of records in R_P that match with records in R_Q. Similarly Q will know only the set Q_{Match}. Of particular importance is that no information will be revealed to P and Q concerning records that do not match each other.

We first solve the problem under hypothesis 1, and then under the more general hypothesis 2. Under hypothesis 1, we provide the secure data matching protocol described in Section 3. Under hypothesis 2, we complement the data matching protocol with the schema matching protocol, described in Section 4, and we provide the overall protocol description in Section 5.

3. SECURE DATA MATCHING

In order to perform secure data matching, pairs of records must be compared by means of comparison functions, typically defined for each attribute. Whichever party performs the comparison, it should compute a distance between attribute values, without seeing the plain values of these attributes. Nevertheless, it is not possible to simply encrypt values, as encryption does not generally preserve the distance between values.

In order to solve this problem, besides the two parties P and Q, we introduce a third party, named W. The three parties are assumed to be semi-honest [7], meaning that they follow the protocol, but they are also allowed to try inferring information. Our approach is based on the idea of transforming records of R_P and R_Q into a metric space while preserving the distances between the record values. Such transformation has the main purpose of protecting data from privacy disclosure. The third party W will compare the records in the metric space in order to decide their matching. Several methods to embed a set of objects in a metric space have been proposed, including FastMap, Multidimensional Scaling, and SparseMap (see [14] for a survey). Among such methods, we have chosen SparseMap, since it meets our goal of preserving privacy, as discussed in Section 6. The following section introduces the basic concepts of this method and motivates the use of this embedding technique with respect to our privacy requirements. Section 3.2 then presents our privacy-preserving data matching protocol.

3.1 SparseMap

SparseMap is a particular instance of a class of embeddings known as Lipschitz embeddings [6]. Therefore, we first describe such embeddings and then we show how SparseMap provides heuristics to make them applicable in practice.

Lipschitz embeddings define a coordinate space where each axis corresponds to a reference set which is a subset of the objects to be embedded. More specifically, given a set O of objects and a distance d in the original space, the embedding is defined in terms of a set S of subsets of O, $S = \{S_1, \ldots, S_k\}$, where each S_i is a *reference set.* Given an object $o \in O$, the mapping F is defined as $F(o) = (d(o, S_1), \ldots, d(o, S_k))$, where $d(o, S_i)$ is the minimum distance $d(o, x)$ for each x in S_i. Therefore, the embedding is such that the coordinates of o are the distances of o from the closest elements in each set. The method is actually based on the triangle inequality and exploits the fact that if $|d(o_1, x) - d(o_2, x)| \leq d(o_1, o_2)$ then, when considering distances from subsets of the type $|d(o_1, S_i) - d(o_2, S_i)|$, we have a lower bound for $d(o_1, o_2)$. This is most likely to occur if the reference sets are randomly selected according to the following properties: *(i)* the number of subsets is proved to be O($log^2 N$) where N is the number of objects, and it is practically approximated by $\lfloor log_2 N \rfloor^2$; *(ii)* each S_i has a size 2^q where $q = \lfloor (i-1)/(log_2 N) + 1 \rfloor$.

The distance metric used to compare the embedded objects, called *vectors*, can be one of the Minkowski metrics, which are metrics based on the norms $\|x\|_p = (\sum |x_i|^p)^{\frac{1}{p}}$ the most popular of which is the Euclidean distance metric, for which p=2. For the purpose of this work, we consider the Euclidean distance metric d^E as the distance metric used in the embedded space. Lipschitz embeddings suffer from two major problems. Firstly, they require O(N^2) distance computations for the embedding of an object o, as the distances between o and all objects of O must be computed. If N is large, the cost of this computation can be prohibitive. Secondly, the number of subsets is also large, being equal to $\lfloor log_2 N \rfloor^2$. As the number of subsets corresponds to the number of dimensions of the space, it should not be too large. A typical dimensionality could be around 30.

HEURISTIC: Distance Approximation

```
Input:  Object $o$, Set $S_i$
Output: Approximate $d(o, S_i)$

(1)  Compute the Euclidean distance $d^E_{i-1}(o, o_j)$ for each $o_j$ in $S_i$;
(2)  Select $o_t$ such that $d^E_{i-1}(o, o_t) = min(d^E_{i-1}(o, o_j))$ for each $o_j$ in $S_i$;
(3)  Return the actual distance $d(o, o_t)$.
```

HEURISTIC: Greedy Resampling

```
Input:  $m$ coordinates
Output: $t <= m$ most discriminating coordinates

(1)  Sample some random pairs $(o_1, o_2)$, and compute their distances $d(o_1, o_2)$;
(2)  Select the reference set $S_i$ that minimizes the STRESS $\sqrt{\frac{\sum_{o_1,o_2} (d^E(F(o_1),F(o_2))-d(o_1,o_2))^2}{\sum_{o_1,o_2} (d(o_1,o_2))^2}}$;
(3)  Repeat $t$ times: add the coordinate axis that leads to the least stress when using in conjunction with the previous selected coordinates.
```

SparseMap overcomes such general limitations of Lipschitz embeddings by means of two heuristics for: *(i) distance approximation*, which approximates the distance between a point o and a subset S_i and *(ii) greedy resampling*, which keeps only the best (most discriminating) coordinates. The pseudo-code of the two heuristics is just above.

SparseMap was originally proposed for the mapping of a database of proteins in the Euclidean space. The application of the heuristics reduced the cost of building the space at the price of a decreased quality of the embedding. We have used a variant of SparseMap that ensures *contractiveness* of the embedding, as proposed in [14]. Contractiveness guarantees that distances in the embedded space are a lower bound for distances in the original space, thus improving the quality of the embedding in terms of recall, i.e., the proportion of correct results found in the embedded space. However, in general, it can be the case that the tradeoff between the usage of the heuristics and the quality of the embedding has to be appropriately considered, as we will further discuss in the experimental section.

3.2 Data Matching Protocol

In this section, we describe the steps of the data matching protocol that solves our problem statement under the hypothesis 1. That is, we assume that the relations owned by the two parties P and Q have the schemas $R_P(A_1, \ldots, A_n)$ and $R_Q(A_1, \ldots, A_n)$. We propose a third party-based protocol that consists of the three following phases.

Phase 1: Setting of the embedding space. Phase 1 consists of the steps 1, 2 and 3 outlined in Figure 1.a. In this phase, P and Q agree on a set of random strings to be used for the generation of the embedding space and on the distance `dist` to use as a comparison function[1]. Note that P and Q start by generating random strings, so not yet using the data from their own datasets. The set of strings is built as follows:

- P and Q agree on a length for the strings to be generated, say L_x, and on the total number of strings to be used for the embedding, say N (see step 1 in Figure 1.a).
- P (or alternatively Q) generates a set G of N random strings of length L_x (see step 2 in Figure 1.a).

In Section 7, we will see how the number N of strings can be appropriately chosen in order to have a high quality of the resulting embedding. The distance metric `dist` is also agreed upon by P and Q in order to build the embedding space. At this point, P (or alternatively Q) builds the embedding space by applying the SparseMap method according to the following steps (see step 3 of Figure 1.a):

- The reference sets are built starting from strings in `G`. Recall that the number of sets and the size of each set is fixed, while each set includes randomly chosen strings. Let $S_1, \ldots, S_z$ be the identified reference sets.
- For each string `s` in `G`, the corresponding coordinates in the embedded space are computed as $dist(s, S_i)$, for each S_i. Let $v_{s1}, \ldots, v_{sN}$ be the vectors of z real coordinates that correspond to the strings in `G`.

The result is a set of N vectors corresponding to the embedding of the strings, and a set of reference sets that are the input to Phase 2.

Example 1. In Figure 2.a, two example datasets R_P and R_Q are shown; for sake of simplicity, only one attribute A_1 is considered, and the values are three strings for both parties. Suppose that the sharing step of the protocol leads to the following values: L_x=5, N=7 and `dist`=edit distance. The random generation of 7 strings allows the construction of the reference sets shown in Figure 2.b. The are 4 reference sets (equals to $O(\lfloor log_2 7 \rfloor^2)$); the corresponding vectors in the space are also shown in Figure 2.b; for instance, the embedded vectors corresponding to S_1 are computed as follows:

- $(d(\texttt{BpuVH}, S_1), d(\texttt{BpuVH}, S_2), d(\texttt{BpuVH}, S_3), d(\texttt{BpuVH}, S_4))$ = `(0.0,5.0,4.0,5.0)`;
- $(d(\texttt{wyZrT}, S_1), d(\texttt{wyZrT}, S_2), d(\texttt{wyZrT}, S_3), d(\texttt{wyZrT}, S_4))$ = `(0.0,5.0,5.0,0.0)`.

Phase 2: Embedding of R_P and R_Q values. Phase 2 consists of the steps 4, 5 and 6 shown in Figure 1.a. First, P performs the embedding of its own dataset by applying the two SparseMap heuristics described in the previous section. The application of the greedy resampling heuristic reduces the space dimensionality from z to k (see step 4 in Figure 1.a). Then, P embeds each value of each record in the Euclidean space; a vector in the Euclidean space corresponds to an attribute value in the table R_P. In order to keep track of the semantics of each vector in the space, a structure P_{str} that mimics the R_P table has to be built by P, as shown in Figure 3. In the figure, each $v_{h,s}$ is a vector corresponding to the embedding of the value of the record h on the attribute s. Such values are computed as follows:

- P decomposes R_P by projecting R_P on each column A_i.
- The k reference sets $S_1, \ldots, S_k$ previously computed are considered as the reference sets and, for each separate column, the set of corresponding vectors is computed as usual, that is, by computing the distances $dist(x, S_i)$ where x is a value in $\Pi_{A_i}(R_P)$.

The space built by one of the parties is also appropriate for the embedding of the dataset of the other party, as it is built on datasets that are "similar", because of the purpose of the matching.

Then, P sends Q the $(S_1, \ldots, S_k)$ reference sets (see step 5 in Figure 1.a). Q builds a structure similar to P_{str}, that we denote as Q_{str}. Let us call $w_{h,s}$ the vectors corresponding to the embedding of R_Q values.

Both structures, that is, P_{str} and Q_{str}, are finally sent to W for the comparison in the subsequent phase 3. Notice that in order for W to perform the comparison among the embedded records, a similarity threshold value for each attribute to be compared needs to be fixed. We assume that the threshold values are agreed between the two parties and are fixed for each attribute. Then, they are sent to the third party W for performing the comparison; as shown in Figure 3, they are part of P_{str} and Q_{str}. As we will see in Section 4, such thresholds can be agreed upon by the parties P and Q without revealing the actual attributes, but by establishing the thresholds on a more general schema exported by the third party.

Example 2. The application of the first heuristic, namely distance approximation, to R_P, leads to the vectors shown in Figure 2.c. The first coordinate is calculated directly by

[1] Notice that a special case of comparison function can be the "equality" function that checks exact matching: this makes the protocol appropriate for exact matching too.

finding the minimum distance to S_1, which is the distance between `academic` and `BpuVH`, equal to 8. To obtain the second coordinate, we calculate approximate distances in a partially embedded space. The distances are computed only by making use of the first coordinate; they are: *(i)* d(8.0,first coordinate of first vector of S_2)=d(8.0,5.0)=3; *(ii)* d(8.0,first coordinate of second vector of S_2)=d(8.0,5.0)=3. The minimum distance is 3 as a result either from the first vector or from the second vector of S_2. If we assume to choose the minimum distance as a result of the second vector, we can calculate the actual distance between `academic` and `aepIy`, thus obtaining the second coordinate equal to 5. Therefore, so far our embedding of `academic` has the two coordinates (8.0,5.0). By proceeding with this approach, we apply the heuristics to the computation of distances from S_3 and S_4 as well, and we finally obtain the full vector (8.0,5.0,7.0,7.0) shown in Figure 2.c. A similar procedure is applied to the remaining strings in R_P.

For the application of the greedy resampling heuristic, the purpose of which is to select the best coordinates, we perform the following steps:

1. We pick a sample of records from R_P, for example: *(i)* `academic` (8.0,5.0,7.0,7.0); and *(ii)* `usefull` (6.0,6.0,6.0,7.0).

2. We consider the stress parameter as defined in the pseudo-code of the greedy resampling heuristic (Section 3.1). By using only the first coordinate, the stress would be 0.5625; by using only the second coordinate the stress would be 0.7656; by using only the third coordinate the stress would be again 0.7656; by using only the fourth coordinate the stress would be 1.0. Therefore we choose the first coordinate as the best.

3. We now use the first coordinate in conjunction with another coordinate to detect the two best coordinates which minimize stress. The first and second coordinates together result in a stress of 0.5191; the first and third coordinates together result again in a stress of 0.5191; first and fourth yield even a higher stress of 0.5625. Thus, we can conclude that the best coordinates are either the first and the third, or the first and the second.

The result of the application of the greedy resampling heuristic is shown in Figure 2.d, where both the R_P and R_Q embeddings are shown.

Phase 3: Comparison to decide matching records. Phase 3 is the only phase that actually involves the third party W. The steps performed by W are the following:

- The vectors of P_{str} and Q_{str} are accessed by means of a multidimensional index (see [13] for a survey) [2].

- A nearest neighbor search is applied in order to compare the vectors of P_{str} and of Q_{str}; the used distance metric is the Euclidean distance. Specifically, given a vector v in P_{str} and a vector w in Q_{str} (recall that the dimensionality of the vectors is k) the Euclidean distance between v and w is:

$$d^E(v,w) = \sqrt{\sum_{i=1\dots k}(v_i - w_i)^2}$$

- Each record in P_{str} is compared with each record in Q_{str}. For the records p_1 in P_{str} and q_1 in Q_{str}, the following decision rule is applied:

$$((d^E(v_{11},w_{11}) \leq \theta_1) \bigwedge (d^E(v_{12},w_{12}) \leq \theta_2) \bigwedge$$

$$\dots \bigwedge ((d^E(v_{1n},w_{1n}) \leq \theta_n))$$

  If the rule is true, p_1 and q_1 are inserted in two sets, called P_{Match} and Q_{Match}, respectively. Similar decision rules are applied for all the record comparisons.

- The final sets P_{Match} and Q_{Match} are sent to the two parties separately. P_{Match} and Q_{Match} are two structures similar to P_{str} and Q_{str}, but containing only the subset of the actual matching records.

Example 3. By computing the (normalized) distances between the obtained vectors in the embedding space, the third party W generates the following matching results:

- The nearest neighbor of record 1 of R_P is record 3 of R_Q, with normalized distance 0.923. This is a false positive as `academic` is not the same as `useful`.

- The nearest neighbor of record 2 of R_P is record 2 of R_Q, with normalized distance 0.858. This is a true positive.

- The nearest neighbor of record 3 of R_P is record 3 of R_Q, with normalized distance 1. This is a true positive.

4. SECURE SCHEMA MATCHING

In this section, we adopt hypothesis 2: the two parties can store records to be matched in a different way so that an integration step must be performed. We propose a schema matching protocol suitable for a general data integration scenario, that involves the third party W for providing a global schema on which the two parties P and Q map their own local schemas. The final result of the protocol is the computation of the set of the *common* attributes used by P and Q for storing their respective records.

Let S_W be the global schema owned by W expressed in a language L_W over an alphabet α_W. Let S_P and S_Q the source schemas owned by the two parties; each schema can be in general expressed in a language different from L_W. The protocol does not reveal to P any schema information about S_Q, and similarly it does not reveal to Q any schema information about S_P. The protocol only reveals to the third party the number of attributes that are common to S_P and S_Q (see Section 6).

Each party maps its local schema to the global schema owned by W. Specifically, each party defines mapping expressions that specifies how S_P (respectively S_Q) elements are related to S_W elements. For instance, if S_W is a schema `Customer(Name,DateofBirth,ResidenceAddress)` and S_P is a schema `Cust(FirstName,LastName,DateofBirth)`, then an example of mapping expression is given by

[2] In the experiments, we have used the KDTree [13] index, as it is considered one of the most prominent data structure for indexing multidimensional spaces; the index is used when comparing a vector against a set of possible matching vectors.

`concatenate(Cust.FirstName,Cust.LastName) =Customer.Name` [22]. By applying the mapping expressions, each party obtains a matched schema, S'_P and S'_Q respectively, which is expressed in the global schema language L_W.

Protocol 1: Privacy Preserving Schema and Data Matching

```
Input: $S_P$,$S_Q$
Output: $P_{Match}$,$Q_{Match}$

(1)  P generates $S'_P(D_1,\ldots,D_s)$ from the mapping of $S_P$ with $S_W(D_1,\ldots,D_L)$;
(2)  Q generates $S'_Q(D_1,\ldots,D_x)$ from the mapping of $S_Q$ with $S_W(D_1,\ldots,D_L)$;
(3)  P and Q negotiate: secret key k; set of threshold values $\{\theta_1 \ldots \theta_L\}$; embedding parameters (string length $L_x$, number of strings $N$, comparison function dist);
(4)  P sends $H_P = (h(D_1,k),\ldots,h(D_s,k))$ to W;
(5)  Q sends $H_Q = (h(D_1,k)\ldots,h(D_x,k))$ to W;
(6)  W computes the intersection $H_P \cap H_Q = (h(D_1,k),\ldots,h(D_i,k))$;
(7)  P constructs the embedded space by: generation of a set G of N random strings of length Lx; building the reference sets $(S_1,\ldots,S_z)$;
(8)  P embeds $S'_P(D_1,\ldots,D_s)$ and performs dimensionality reduction from z to k;
(9)  P sends $S_1,\ldots,S_k$ to Q;
(10) Q embeds $S'_Q(D_1,\ldots,D_x)$;
(11) P constructs $P_{str}$ by key-hashing attributes of the schema with k;
(12) P sends $P_{str}$ to W;
(13) Q constructs $Q_{str}$ by key-hashing attributes of the schema with k;
(14) Q sends $Q_{str}$ to W;
(15) W computes the intersection $P_{str} \cap Q_{str}$, on the values of the attributes in $H_P \cap H_Q$;
(16) W builds $P_{match}$ and $Q_{match}$ including matching records padded with attribute values not involved in the matching;
(17) W sends $P_{match}$ to P; W sends $Q_{match}$ to Q.
```

In this way, W has the possibility of intersecting the sets of attributes of S'_P and S'_Q. We use the universal relation approach [18] to model data at the third party W: the third party exports its global schema in the form of a unique global relation $S_W(D_1,\ldots,D_L)$; given the set of attributes of the global relations, the schema exported by W is the union of all attributes of all global relations. The matched schema S'_P and S'_Q will thus consist each of a single relation $S'_P(D_1,\ldots,D_s)$ and $S'_Q(D_1,\ldots,D_x)$ respectively.

The protocol consists of the steps shown in Figure 1.b. First, P and Q negotiate a secret key k unknown to W (step 1). The purpose of the key is to avoid privacy breaches by the third party, possible in a honest-but-curious context. Then, both P and Q *map* their own schemas onto $S_W(D_1,\ldots,D_L)$ by *(i)* designing the mapping expressions and *(ii)* generating the matched schemas $S'_P(D_1,\ldots,D_s)$ and $S'_Q(D_1,\ldots,D_x)$ (see steps 2.a and 2.b in Figure 1.b). Besides publishing S_W, the third party also publishes a hash function `h`, accessible to both P and Q. This function is used in conjunction with the exchanged key, by both P and Q, to encrypt their own attribute names. Specifically, P and Q use keyed-hashing to encrypt the attributes of the schemas S'_P and S'_Q. Therefore, P generates the set $H_P = (h(D_1,k),\ldots,h(D_s,k))$, Q generates $H_Q = (h(D_1,k)\ldots,h(D_x,k))$, and then both parties send such encrypted schemas to W (see steps 3.a and 3.b of Figure 1.b). Finally, W computes the intersection $H_P \cap H_Q = (h(D_1,k),\ldots,h(D_i,k))$ (step 4).

Notice that for the purpose of our problem statement, it is not necessary for the third party W to send the result of the intersection to the two parties P and Q. However, the schema matching protocol could be also executed independently on the instance matching one, and in that case the set of matching attributes is sent to the parties that can decrypt it on the basis of the known key k.

Discussion. The hypothesis of the third party publishing the global schema independently of the sources is the same performed by several concrete standardization initiatives. Current approaches to Business-to-Business (B2B) [20, 24] and eGovernment [25] are based on the definition of *a-priori* catalogues of standard processes and data schemas to be used for data and process integration on the Internet. Clearly, the philosophy underlying such efforts is that: *(i)* commonly agreed upon reference schemas (either for processes or for data – the focus of our work) can be defined in specific vertical sectors without knowing the exact details of the possible organizations adopting them in the future; *(ii)* when specific organizations adopts them, they only need to map and convert their internal systems to the predefined schemas, without a preliminary phase in which such global schemas are negotiated among the participating organizations. During this mapping phase, as organizations operate in a coopetitive environment, they do not want to expose to others the internals of their systems. This is also why the standards are a-priori predefined, in order not to be biased towards any organization possibly adopting them in the future.

5. OVERALL PROTOCOL

In this section, we illustrate the pseudo-code of the overall protocol solving the problem statement under the more general assumptions of hypothesis 2 (see Protocol 1). The protocol consists of the sequential application of the schema matching protocol and of the data matching protocol.

The following modifications to the data matching protocol are required:

- The relations that are input to the instance matching part of the protocol are $S'_P(D_1,\ldots,D_s)$ (line 8) and $S'_Q(D_1,\ldots,D_x)$ (line 10). Notice that the information on the common attributes $D_1,\ldots,D_i$ is owned only by the third party (line 6).

- The two parties P and Q agree on a set of threshold values $\theta_1,\ldots,\theta_L$ related to the global schema $S_W(D_1,\ldots,D_L)$. Then, the third party actually uses only the threshold values needed for comparing the common attributes.

- The structure P_{str} (and similarly Q_{str}), which is shown in Figure 3, will have the attribute names properly encrypted with the key shared by the two parties in the step 1 of the schema matching protocol (first row in Figure 3), in order to prevent W from seeing such names in plain.

- The results P_{match} and Q_{match} are *recomposed* by the third party before sending them back, by adding also attributes not common to the parties (and therefore not used in the matching).

The final outcome of the protocol for a party is the disclosure of which of its own records are shared by the other party too. This scenario implies that each party knows the status of matching of one's own data set with the one of the other party, but no private data are revealed at all. We plan to investigate how *merging* matched records as a further result of the matching process in future work.

6. SECURITY ANALYSIS

In this section, we analyze the security properties of the protocol described in the previous sections. The elements involved in the security analysis are: length of database records (`Len`), database size (`DBsize`), set of matching records (`RecMatch`), set of matching attributes (`AttrMatch`), and number of matching attributes (`AttrMatchSize`).

The number of matching attributes (`AttrMatchSize`) is an element the privacy of which needs to be controlled, in that it is a degree of how much the matching entity representations by the two parties are similar. Conversely, the number of matching records is not an element that needs to be controlled as it is an output of the protocol (being the set of matching records revealed to the two parties). In the protocol, there are some steps in which though exact elements are not revealed, a relationship exists between them and other known elements. Specifically, given such relationships, an upperbound (or lowerbound) estimation of the elements can be performed. Therefore, we distinguish three different values for the disclosure of a given element: *(i)* disclosed; *(ii)* bounded and *(iii)* not disclosed.

The following lemmas state the behavior of the protocol with reference to the above parameters. Recall that S'_P and S'_Q are the relations owned by the two parties that are subject to the instance matching protocol.

LEMMA 1. ***Len*** *of S'_P and S'_Q is:* (i) *not disclosed to the third party W;* (ii) *bounded by the two parties P and Q.*

PROOF. (Sketch) *(i)* follows directly by the fact that W is not involved in the step of building the embedding space. *(ii)* derives from the fact that the length of both S'_P and S'_Q records can be bounded by the length L_x of the strings used for the embedding construction. □

LEMMA 2. ***DBSize*** *of S'_P and S'_Q is:* (i) *disclosed to the third party W;* (ii) *bounded by the two parties P and Q.*

PROOF. (Sketch) *(i)* follows directly from the fact that the third party W can derive `DBSize` of S'_P and S'_Q from P_{str} and Q_{str} which have exactly the same size of S'_P and S'_Q (lines 12 and 14 of the protocol). *(ii)* follows by observing that `DBSize` is bounded by the parties because it depends on the number N of random strings used to generate the embedding. □

LEMMA 3. ***AttrMatchSize*** *of S'_P and S'_Q is:* (i) *disclosed to the third party W;* (ii) *not disclosed to the two parties P and Q.*

PROOF. (Sketch) The size of the intersection $(D_1, \ldots, D_s) \cap (D_1, \ldots, D_x)$, that is, `AttrMatchSize`, is disclosed to the third party that actually computes the intersection between their encrypted value (line 6 of the protocol). On the other hand, given the protocol steps, W is in charge of *recomposing* P_{str} and Q_{str} for the purpose of not revealing `AttrMatchSize` (line 16 of the protocol), thus proving *(ii)* . □

On the basis of the above lemmas, we state the following theorem on the security of the schema matching protocol proposed in the paper.

THEOREM 1. *Given the schemas S'_P and S'_Q, owned by parties P and Q respectively, the schema matching protocol finds the attributes common to the two schemas with the following assurance:* (i) ***AttrMatch*** *is not disclosed to W;* (ii) ***AttrMatch*** *is not disclosed to P and Q;* (iii) ***AttrMatchSize*** *is not disclosed to P and Q.*

PROOF. *(i)* follows by the use of key-hashing for encrypting the attributes to be sent to W (see lines 4, 5, 11, 13). *(ii)* is a consequence of the protocol steps (the intersection is not sent to the parties by W) and *(iii)* follows from the Lemma 3. □

As far as the security of the data matching protocol, the following theorem holds.

THEOREM 2. *Given the two relations $S'_P(D_1, \ldots, D_s)$ and $S'_Q(D_1, \ldots, D_x)$, owned by parties P and Q respectively, the data matching protocol finds the matched records between the two relations with the following assurance:* (i) ***RecMatch*** *is not disclosed to W;* (ii) *$S'_P -$* ***RecMatch*** *is not disclosed to Q and $S'_Q -$* ***RecMatch*** *is not disclosed to P;* (iii) ***DBSize*** *is disclosed to W and bounded by P and Q.*

PROOF. (Sketch) *(i)* is proved by considering that W cannot infer the plain values of records of S'_P and S'_Q, as the third party only sees the vector representation of such values. This representation is the result of some protocol steps that make unfeasible for the third party to learn the original values, namely: generation of the embedding on the basis of random strings, the number and length of which are only known by P and Q (see lines 1 and 7 of the protocol); random choice of the reference sets for the embedding (see line 7 of the protocol and Lemma 1); heuristics-based selection of the vector space coordinates (see line 8 of the protocol). *(ii)* is proved by the fact that both P and Q never obtain the data sets of the other party during the protocol, neither in a plain representation nor in the embedded representation. *(iii)* follows from Lemma 2. □

7. EXPERIMENTS

We performed an extensive experimental evaluation of our protocol, with three distinct goals:

- Finding the parameters necessary for building the embedded space and testing the quality of the embedding. We experimentally derived several embedding parameters and we analyzed the distortion of the embedding resulting from the modified version of the SparseMap method that we used.

- Illustrating the effectiveness of the data matching protocol. We analyzed the effectiveness of the record matching in terms of the classical recall and precision metrics and we compared the obtained results with respect to the same record matching performed in the original space.

- Measuring the efficiency of the data matching protocol. We compared our method to the time performance of the secure protocol proposed by Agrawal et al. [2] which, differently from our method, performs exact matching. Finally, we also show the time performance of our method when compared to the record matching performed in the original space.

We used three real datasets to evaluate our protocol. The first dataset, referred to as dataset 1, represents a British Columbia voters' list (available at `http://www.rootsweb.com/∼canbc/vote1875/`). From such data set, we extracted first and last names, and generated two data sets of 1000 records each. The dataset contains roughly 500 duplicate records (errors in these duplicates were introduced synthetically). The second dataset, referred to as dataset 2, contains personal data by a public administration. It consists of two tables of sizes 7846 and 7550 respectively, with 6684 duplicates already identified. The fields used for the experiments are first name, last name, and the year of birth of individuals. The third dataset, referred to as dataset 3, contains information about businesses and is owned by a public administration. The size of this data set is about 20.000 records. The duplicates of this data set were artificially generated as it has been mainly used to test the efficiency of the method, by extracting from it several data samples of different sizes.

We conducted out experiments on a 3.00GHz Pentium 4 Processor running Windows XP, with a RAM of 1GB. We chose Java 1.5 to implement the experiments, and to leverage open source libraries available online. In particular, we used Secondstring library (available at `http://secondstring.sourceforge.net/`) that provides a variety of similarity metrics. For indexing the embedded space we used a KDTree implementation (available at `http://www.cs.wlu.edu/∼levy/kd/`). We also used Java Cryptography Extension (available at `http://java.sun.com/products/jce/`) for implementing the exact private matching protocol described in Agrawal et al. [2].

7.1 Building the Embedding Space

As discussed in Section 3, our method builds the embedded space starting from a set of randomly generated strings, in order to meet the privacy requirements of our protocol. In this section, we illustrate a set of experiments that shows how to set the parameters necessary for building the embedded space in order to obtain high quality of the resulting embedding.

We have considered how to choose the number and the size of the random strings necessary to build the reference sets of the embedding. We recall that the relationship between the number of random strings N_s and the dimensionality d of the space is such that d is approximated by $(log_2(N_s))^2$. Therefore, N_s can be easily determined by an appropriate choice of the dimensionality. We evaluate the quality of the embedding with respect to the *stress* parameter (as defined in the greedy resampling pseudo-code in Section 3), measuring the distortion of the embedding, and with respect to the recall and precision metrics. We report the results obtained for dataset 1.

Figure 4.a reports stress against several dimensionalities of the space for different string lengths. As expected, the lowest stress value is reached in correspondence of higher dimensionality values. Increasing the string length also results in a lower distortion; for data set 1 the average length of the strings to be embedded is about 20. On the basis of further experiments performed on dataset 2, we concluded that a general guideline is to use random strings of length which is approximately equal to the length of elements in the data set to be embedded. Moreover, for dataset 2, which includes the numeric field birth of date, we experimented the usage of random generated strings composed by numbers and alphabetic characters, that resulted in lower stress values. Therefore, it is appropriate to generate random strings sharing a similar alphabet with the data to be embedded.

In Figure 4.b, a second experiment shows how precision and recall values vary depending on the dimensionality for different string lengths. Recall values do not vary, instead precision increases with dimensionality values and with string lengths. On the basis of such experiments, for dataset 1 the chosen embedding parameters are: string length equal to 20, and dimensionality equal to 20 (corresponding to a number of random strings equal to 23).

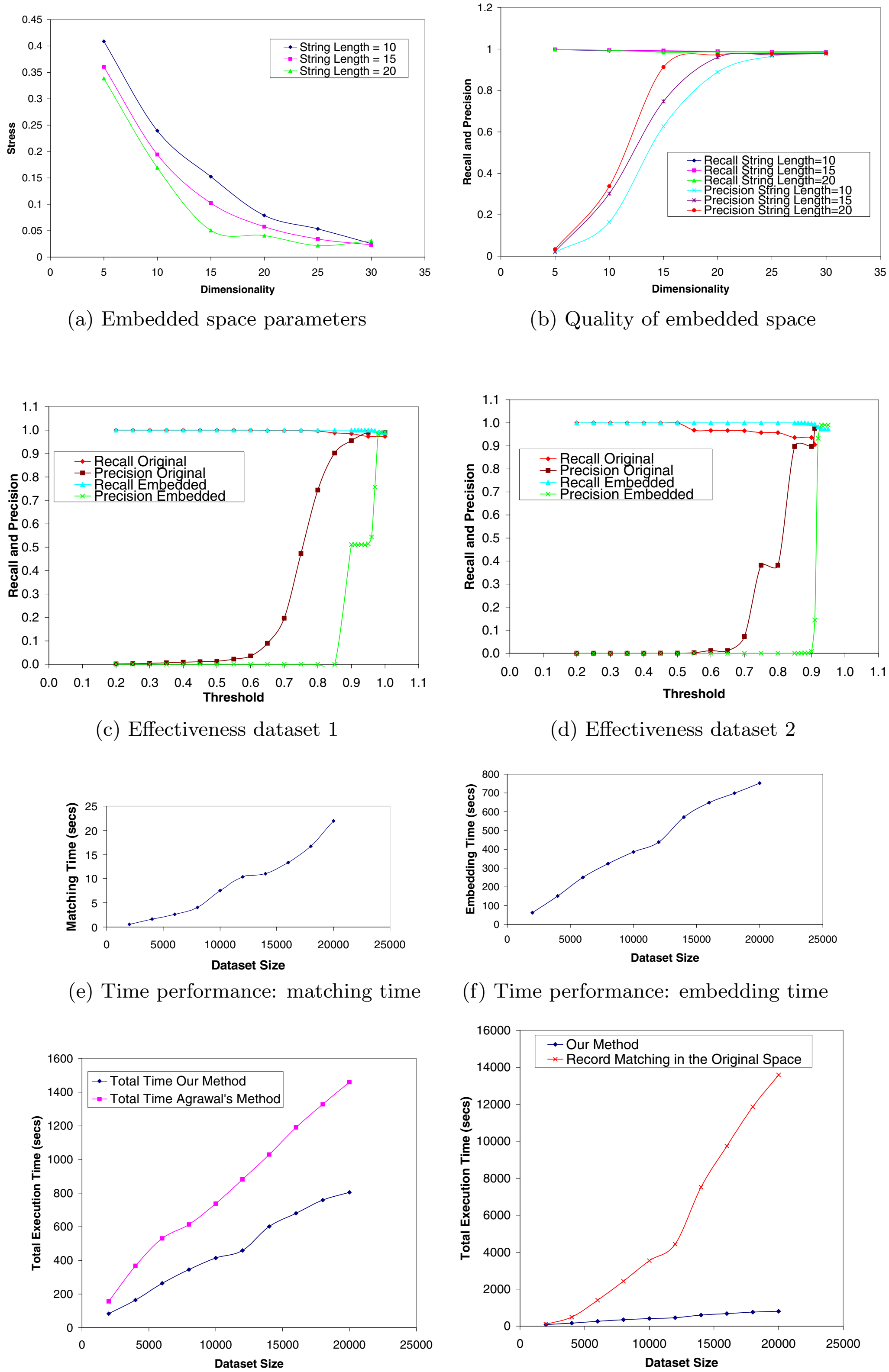
7.2 Testing the Effectiveness of the Data Matching Protocol

We measured the effectiveness of the data matching protocol by comparing precision and recall of our method, which operates in the embedded space, to precision and recall obtained in the original space. We performed the effectiveness experiments on datasets 1 and 2; for both datasets we knew which records were true matches, therefore we could compute recall and precision. The initial experiments were performed with the greedy resampling heuristic disabled, but a final one shows its impact on recall and precision.

The results for dataset 1 are shown in Figure 4.c, in which precision and recall are drawn when varying the similarity threshold. Recall values are very similar in the two cases. This is actually a predictable behavior as the mapping we perform is *contractive*, which implies that the distances in the embedding space lower-bound the corresponding distances in the original space. Precision starts to increase for threshold values in the original space which are lower than the ones in the embedded space. Intuitively, this behavior can be also motivated by contractiveness, which forces to have more discriminating thresholds, i.e., with higher values. However, precision of our method reaches very high values.

The results for dataset 2 are shown in Figure 4.d. The property for precision to reach good values with lower threshold in the original space is still valid. What is interesting to note is that if we compare Figure 4.c and Figure 4.d, the relative difference between the two precision curves in the original and in the embedded space is very similar for the two datasets.

We have also tested the precision and recall performance when applying the greedy resampling heuristic to dataset 2. In such a case, we actually apply the full SparseMap with the variant ensuring contractiveness, so high values for recall



(a) Embedded space parameters

(b) Quality of embedded space

(c) Effectiveness dataset 1

(d) Effectiveness dataset 2

(e) Time performance: matching time

(f) Time performance: embedding time

(g) Time performance vs. Agrawal's method

(h) Time performance vs. record matching in the original space

Figure 4: Experiments on the embedding space and on the effectiveness and efficiency of the data matching protocol

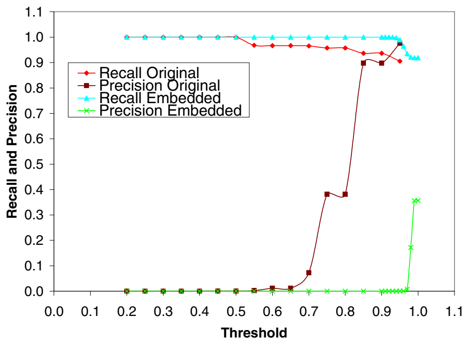


Figure 5: Effectiveness dataset 2 with greedy resampling

are still guaranteed. However, as shown in Figure 5, the precision is much lower, not reaching 0.4. This result suggests a careful use of the heuristics by balancing the tradeoff between a more efficient building of the space and its quality.

7.3 Testing the Time Efficiency of the Data Matching Protocol

The efficiency experiments were performed on data set 3 by considering samples of different sizes.

We identified three different time features to evaluate, namely: *(i)* indexing time, that is, the time to build the index; *(ii)* matching time, that is, the time to perform the actual comparison and matching decision and *(iii)* embedding time, that is, the time to build the embedding space. The indexing time is not significant when compared to the other two times. In Figure 4.e, we report the matching time for varying values of the data set dimension. The time increases linearly with the size of the data set, and for comparing 20000 records such time is lower than 25 seconds. The time necessary for embedding samples of different sizes of dataset 3 is shown in Figure 4.f. Such time also exhibits a good behavior being approximately linear.

We have compared such time performance with the time performance of the algorithm proposed by Agrawal et al. in [2] (called in the following Agrawal's method). This algorithm performs a privacy preserving intersection of two data sets, by assuming an *exact* matching of data. Our method is more general in that it allows an approximate matching. However, the purpose of the comparison is to show that better time performances can be obtained without the usage of cryptographic techniques. In order to perform such a comparison, we coded Agrawal's method by using SHA-512 hash function and an implementation of Pohilig-Hellman commutative encryption scheme. We considered a total time for the execution of the Agrawal's method, by taking into account two major components, namely the encryption time and the ordering time. In Figure 4.g, we compare the total time of our method with Agrawal's method total time. The total time of our method results from the sum of the embedding time and of the matching time. Though our algorithm performs approximate matching, the fact that we do not use any cryptographic technique allows us to achieve twice better time performance than Agrawal's method.

Finally, we have compared the total time of our method with the time required to perform a record matching in the original space. Record matching requires N^2 comparison, being N the size of each dataset to compare. We have considered this worst case performance, though we are aware that such number of comparisons is typically reduced to $O(KN)$ (with $K << N$). This is because our objective is not to propose a more efficient record matching method, but a privacy preserving one. In Figure 4.h, we report the total time of our method and of the record matching in the original space. Notably, we do obtain an efficient record matching without any additional effort.

8. RELATED WORK

The record matching problem (also known as record linkage) has been studied for more than five decades (see the recent survey [10]). However, few methods for *private* record matching have been investigated. Some initial approaches are motivated by the strict privacy requirements of eHealth applications [21, 8]. The work which is closely related to ours is by Al Lawati et al. [4], that propose a third party based protocol. Their solution is based on: *(i)* the usage of the TFIDF (term frequency-inverse term frequency) comparison function, by reducing it to a scalar product of vectors which can be computed separately by each party; *(ii)* a secure blocking scheme. The approach has the disadvantage to work only for a specific comparison function; also, as the focus is mainly on efficiency, the effectiveness of the approach has not been assessed. Conversely, our approach can be used with different comparison functions, it has a proved effectiveness and guarantees privacy preservation also at schema level.

In addition to the above approaches, there are three major areas that are closely related to our work, even though no work in such areas addresses our exact problem: secure set intersection, secure string comparison, private data sharing. Several approaches have investigated the secure set intersection problem (see [12] and [16] for a survey). Secure set intersection methods deal with exact matching and are too expensive to be applied to large databases due their reliance on cryptography. Furthermore, these protocols deal with the intersection of sets of simple elements, and are not designed for exploiting the semantics behind database records. The latter is an important feature of our protocols. The problem of securely comparing strings has been addressed by homomorphic encryption schemes, characterized by the property that $E(a)*E(b) = E(a+b)$. For example Atallah et al. [5] propose an algorithm for comparing sequences based on such schemes. While such algorithm works for sequence similarity, like DNA sequence comparison, the communication cost of this algorithm, proportional to the product of the sequence lengths, is prohibitive for databases, in that it would require for a database of n records to apply the protocol to compare each attribute value of each of the n records. Ravimukar et al. [23] propose a method to compute distance metrics securely, when they can be expressed in terms of weights to be computed separately by the parties, like TFIDF. Because such solution relies on cryptographic techniques, it is not efficient and appropriate for databases. Agrawal et al. [2] formalize a general notion of private information sharing across databases. Such notion relies on commutative encryption techniques and has opened the way to other related protocols [11, 1]. These approaches, however, do not deal with approximate matching in that they focus on exact string matching for the purpose of query answering.

In addition, in Section 7 we have proved the superior time performance of our data matching protocol when compared to Agrawal et al. [2]. In [26, 3], there is the investigation of possible malicious behaviors by the parties involved in an information sharing setting in which only necessary information must be disclosed. From the data management area in the data mining community, there are several contributions regarding how to preserve privacy in distributed contexts [9]; such works are mainly based on perturbing the original data and at the same time achieving correct data mining results.

9. CONCLUSIONS

The paper describes a protocol for privacy-preserving record matching between two parties that can have different schemas and privacy requirements also at schema level. The protocol does not rely on complex cryptographic techniques that are proven to be inefficient. Instead, it exploits the novel idea of obtaining privacy by embedding the records of each of the two parties in a vector space. We have proved the security of the protocol by a detailed analytical analysis and its effectiveness and efficiency by means of experimental validation. The experimental evaluation has shown the possibility to reach an almost 100% recall, and very high levels of precision that are comparable with the ones achieved when performing record matching in the original space. Moreover, the time performance experiments have shown the superiority of our protocol when compared to methods that rely on cryptography for privacy preservation. We believe that our protocol could pave the way for more efficient secure set intersection protocols for database applications.

Future work will address more complex data integration scenarios. First, we will consider scenarios with more than two parties. In this extension, there are several issues that should be carefully designed and evaluated; for instance, how to optimize the step of building the embedding space. Various kind of assumptions can also be made on the behavior of the parties involved in the protocol, such as introducing possible malicious behaviors by some of the parties. Moreover, we will investigate the possibility of using embedding methods different from SparseMap in order to evaluate their suitability for privacy preserving record matching, and compare them with the results obtained with SparseMap (for instance, in terms of the complexity for building the embedded space and the resulting quality of the space itself). Finally, an important feature of our protocol is that it can be applied to database with novel data types like images, DNA sequences, proteins, etc. We plan to investigate the actual performance of our protocol when applied to these data types.

Ackowledgements. This work was supported by the US grants NSF-0430274 and NSF-ITR 0428168, a Lilly Endowment grant, a NAVSEA/NSWC CRANE grant, a US DHS (PURVAC) grant, by the sponsors of CERIAS, by the Italian MIUR project ESTEEM and by the EU ISTAT CENEX-DI.

10. REFERENCES

[1] R. Agrawal, D. Asonov, M. Kantarcioglu, Y. Li, *Sovereign Joins*, ICDE 2006.

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

[3] R. Agrawal, E. Terzi, *On Honesty in Sovereign Information Sharing*, EDBT 2006.

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

[5] M.J. Atallah, F. Kerschbaum, W. Du, *Secure and Private Sequence Comparisons*, WPES 2003.

[6] J. Bourgain, *On Lipschitz Embedding of Finite Metric Spaces in Hilbert Space*, Israel Journal of Mathematics **52** (1985), no. 1-2, 46–52.

[7] R. Canetti, U. Feige, O. Goldreivh, M Naor, *Adaptively Secure Multi-party Computation*, STOC 1996.

[8] T. Churces, P. Christen, *Some Methods for Blindfolded Record Linkage*, BMC Medical Informatics and Decision Making **4** (2004), no. 9.

[9] C. Clifton, M. Kantarciougluu, X. Lin, J. Vaidya, M. Zhu, *Tools for Privacy Preserving Distributed Data Mining*, SIGKDD Explorations **2** (2003), no. 4, 28–34.

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

[11] F. Emekci, D. Agrawal, A. El Abbadi, A. Gulbeden, *Privacy Preserving Query Processing using Third Parties*, ICDE 2006.

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

[13] V. Gaede, O. Gunther, *Multidimensional Access Methods*, ACM Computing Surveys **30** (1998), no. 2.

[14] G.R. Hjaltason, H. Samet, *Properties of Embedding Methods for Similarity Searching in Metric Spaces*, IEEE TPAMI **25** (2003), no. 5.

[15] L. Jin, C. Li, S. Mehrotra, *Efficient Record Linkage in Large Data Sets*, DASFAA 2003.

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

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

[18] D. Maier, J.D. Ullman, M.Y. Vardi, *On the Foundations of the Universal Relation Model*, ACM TODS **9** (1984), no. 2, 283–308.

[19] M. Naor, B. Pinkas, *Oblivious Transfer and Polynomial Evaluation*, STOC 1999.

[20] OASIS, *e-Commerce Technical Committee*, `http://www.oasis-open.org/`.

[21] C. Quantin, H. Bouzelat, F. Allaert, A. Benhamiche, J. Faivre, L. Dusserre, *How to Ensure Data Security of an Epidemiological Follow-up: Quality Assessment of an Anonymous Record Linkage Procedure*, Int. Journal of Medical Informatics **49** (1998), no. 1.

[22] E. Rahm, P.A. Bernstein, *A Survey of Approaches to Automatic Schema Matching*, VLDB Journal **10** (2001), no. 4, 334–350.

[23] P. Ravikumar, W. Cohen, S.E. Fienberg, *A Secure Protocol for Computing String Distance Metrics*, WPSADM 2004 (at ICDM 2004).

[24] RosettaNet, `http://portal.rosettanet.org/cms/sites/RosettaNet/`.

[25] UK Gov Talk, e-GIF, `http://www.govtalk.gov.uk/schemasstandards/egif.asp`.

[26] N. Zhang, W. Zhao, *Distributed privacy preserving information sharing*, Proc. VLDB 2005.