JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 5, No. 3, September 2011, pp. 223-235

Secure Blocking + Secure Matching = Secure Record Linkage

Alexandros Karakasidis*
Department of Computer and Communication Engineering, University of Thessaly, Volos, Greece **akarakasidis@inf.uth.gr**

Vassilios S. Verykios
School of Science and Technology, Hellenic Open University, Patras, Greece **verykios@eap.gr**

Abstract

Performing approximate data matching has always been an intriguing problem for both industry and academia. This task becomes even more challenging when the requirement of data privacy rises. In this paper, we propose a novel technique to address the problem of efficient privacy-preserving approximate record linkage. The secure framework we propose consists of two basic components. First, we utilize a secure blocking component based on phonetic algorithms statistically enhanced to improve security. Second, we use a secure matching component where actual approximate matching is performed using a novel private approach of the Levenshtein Distance algorithm. Our goal is to combine the speed of private blocking with the increased accuracy of approximate secure matching.

Category: Ubiquitous computing; Security and privacy

Keywords: Security privacy; Record linkage; Approximate matching; Phonetic codes; Edit distance

I. INTRODUCTION

The contemporary era is characterized by a high degree of involvement of computers harvesting data in various aspects of everyday life. From simply carrying a cell phone to traveling and from casual shopping to receiving medical aid, all these activities induce the collection of large volumes of information that also include private data. These independently stored data often need to be integrated.

Combining independently stored information could benefit a large number of domains, such as medical research and public safety. For instance, information integration could lead to the creation of a public safety early warning system. Consider, as an example, the recent H1N1 flu outbreak. Each time a person was found to be infected with the disease, attempts were made to spot his/her recent activities to prevent the spread of the epidemic by locating possible candidate patients. A typical example is the case of airline travelers. When a passenger is found to be infected with the H1N1 virus, a significant effort is made to locate people who traveled on the same flight.

An effort like this would be much more successful if an early warning monitoring system existed that would locate and inform possible patients by integrating data from an airline's passenger list in our case, with the patient list in hospitals. When a traveler is found to be infected with the flu, the other passengers can be instantly alerted. Such a system however would require the preservation of privacy of both patients and travelers. In the case in which the hospital integrated travel and medical data, no information should be revealed to the hospital database administrator regarding names of passengers. Correspondingly, if the airline managed to merge this information, the medical record of the patients should not be fully exposed.

Technically speaking, what we have presented previously is a modified version of the classical record matching problem. In classical record linkage, separate dataholders maintain data corresponding to the same real world entities (people, cases, events, etc.) without necessarily maintaining common and unique linkage identifiers. Dealing with non-unique identification information is an important deficiency that needs to be addressed by all the record linkage algorithms. Additionally, in database fields,

Open Access **http://dx.doi.org/10.5626/JCSE.2011.5.3.223** **http://jcse.kiise.org**

Received 1 February 2011, **Accepted** 20 March 2011
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

such as names, and addresses, typing or spelling errors that may accidentally occur make this task even more difficult. It is obvious that this is a task of increased complexity. As such, a technique called *blocking* has been introduced to improve performance [1]. Private record linkage features an additional major complication during data matching: all data stored within independent repositories contain sensitive individual information.

Therefore, any attempt towards data integration would lead to privacy breaches, raising ethical and legal issues: certain legislation, such as the EU Data Privacy Directive [2], render each company responsible for maintaining the privacy of its stored data and any explicit leaking might lead the dataholder to face prosecution. It is evident that performing data linkage, while preserving privacy is an important problem that needs to be taken seriously into consideration.

To be more concise, the core of the private record linkage problem renders the ability of merging data from two or more databases in a way that the only additional information that each of the database owners gain, will only refer to entities already stored in his own database before integration.

In this paper, we propose a complete framework for addressing the problem of privacy preservation approximate record matching. We achieve this by enhancing Edit Distance based methods used for conventional approximate record matching with privacy-preservation characteristics and increasing matching performance by facilitating a secure blocking component based on phonetic codes. We prove our assertions that our approach offers the same accuracy with traditional edit distance algorithms, while it respects privacy and compare our results to a rival algorithm by extensive experimentation.

In the privacy preservation record linkage protocol we propose, we assume that all parties exhibit an Honest But Curious behavior (HBC). All parties are honest in the sense that they are all obliged to follow the protocol. They are curious, since they maintain every new knowledge acquired during the execution of the protocol. Our protocol satisfies these restrictions, since no new information is revealed, apart from integrating complementary information regarding known real world entities.

The contribution of this paper is multidimensional. First, we propose a complete framework towards secure approximate matching. Second, this framework consists of a secure blocking component and a secure matching component featuring an open architecture. Openness is a very important feature, since we do not provide a single monolithic technique, but a framework which allows the usage of a variety of techniques depending on their suitability in each case. Third, we propose a secure blocking component enhanced by three levels of privacy: phonetic encoding, encryption and random noise insertion in the form of fake phonetic codes. Fourth, we propose an approximate private matching algorithm by enhancing with security the widely used Levenshtein distance algorithm [3], while maintaining intact its matching accuracy. Fifth, we present an in-depth security analysis of our approach. Finally, we provide a set of extensive experiments that prove the correctness, security and high performance of our approach.

The remainder of this paper is organized as follows. Section II presents previous work related to ours. Section III provides necessary background knowledge. Section IV presents the proposed private matching framework. Section V contains the evaluation of our methodology in terms of complexity and privacy. Our approach is empirically assessed in Section VI. Finally, Section VII contains conclusions and future research directions.

II. RELATED WORK

Record matching (or linkage) is a rather old yet important area of research. As such, numerous methods have been proposed to address the problem. A detailed analysis of all major currently used methods can be found in [4]. Approximate string matching methods consider comparing strings to possible typographical errors. These methods fall into three major categories: token-based methods, distance-based methods and phonetics-based methods.

Token-based methods calculate tokens of the strings to be matched and then count the number of common tokens. N-grams based methods fall into this category [5]. Distance-based methods measure the differences between strings. Some of the most widely used methods are Levenshtein distance [3], the Jaro [6] and Jaro-Winkler metrics [7, 8]. Conversely, phonetics-based methods make use of certain string transformations to take advantage of the way words sound for purging the effects of various typing and spelling errors. Typical examples of this class include Soundex [9], Metaphone [10], ONCA [11], and NYSIIS [12]. Based on all the aforementioned methods, a variety of systems, such as Tailor [13], Febrl [14] and WHIRL [15] has been developed to perform approximate record linkage.

Research directions were provided in [16] stating the needs and problems of this research field regarding privacy-preserving matching. State of the art in token-based privacy-preserving matching includes the work of Churches and Christen [17] who propose deidentifying the dice coefficient of n-grams using hashing. However, approaches relying on [17] suffer from very high complexity. Privacy preservation of distance-based methods, such as Edit Distance [18] and the Jaro-Winkler string metric [7] performed well only in cases of non-encrypted data, as indicated in [19]. Previous work in privacy-preserving phonetics-based matching required the existence of a trusted third party [20]. Moreover, as detailed in Trepetin's survey [21], some attempts to address the problem of privacy-preserving record linkage fall short in providing a sound solution, either because they are very resource intensive [22, 23] or because they are unable to provide both matching accuracy and fault tolerance [24].

Approaches that are more recent include the one suggested by Inan et al. [25] that uses a hybrid scheme combining cryptographic methods with anonymization techniques creating value generalization hierarchies. Scanapiecco et al. [26] achieve privacy-preserving data matching using embedding spaces and schema matching based on a global schema. The privacy properties of *k*-anonymity on joins explored by Kantarcioglu et al. can by found in [27]. Guidelines to build a privacy-preserving matching system are detailed in [28] and a categorization of recent approaches in [29]. The latest work in this field is by Inan et al. [30] based on differential privacy.

Atallah et al. [31] proposed an approach that resembles our methodology, where the edit distance algorithm is modified to provide privacy to genome sequence comparisons. This approach

was aimed towards sequence comparisons and has considerable communication cost. In contrast, our methodology has low communication cost, since it targets data stored in files or databases.

Bloom Filters [32] are an essential part of our architecture. They have been used in a variety of contexts, ranging from indexes for XPath queries [33] to summary caches [34]. Schnell et al. [35] proposed a method based on a combination of bigrams and Bloom filters regarding private record linkage. Each matching party creates a Bloom Filter where bigrams are stored. Then, one party manages to compare the filters by performing a logical AND on them and calculating similarity in terms of common bits, using the Jaccard coefficient.

III. BACKGROUND

In this section, we provide the necessary background required to present our methodology, along with a running example used throughout the rest of our paper. Specifically, we describe the phonetic algorithms and distance-based matching methods. Moreover, we present the operation of Bloom filters and their extension, counting Bloom filters used in our approach, and discuss guidelines for selecting the appropriate hash functions for our framework.

A. Example Scenario

Let us go back to the example presented in the introduction, where we wish to integrate a hospital's database with the database of an airline to locate possible H1N1 patients. We consider two parties, A that will represent the hospital database (Table 1) and B that will represent the airline database (Table 2). These two parties will also be referred to as sources.

Both databases hold sensitive information that can be used to uniquely identify individuals. For simplicity, we will assume that both databases have different schemas but share a common subset of attributes, i.e., Name, Lastname, and Area. These data will be referred to as private data, because they should not be revealed. These fields will be used for record linkage as well. Additionally, each of the two databases holds quasi-identifiers. These are data that are not self-descriptive of real world entities, but provide private information if combined with the private data. Let us assume for the hospital that this is the *Room* field and for the airline the *Flight* field.

Table 1. Hospital data (source A)

Row	Name	Lastname	Area	Room
1	Mayi	Pounder	Clarkson	199
2	Cooper	Moble	Kirrawee	054
3	Ryan	Joshua	Andrews	222

Table 2. Airline data (source B)

Row	Name	Lastname	Area	Flight
1	Rayn	Joshua	Andrews	OX532
2	Maya	Pounder	Clarkson	AZ117
3	Iachlan	Grossman	Cranbourne	AAA512

Our aim is to integrate these two databases in such a way that neither the hospital will be aware of all of the airline's passengers, nor the airline will get any knowledge regarding the hospital's patients. The only knowledge that will be gained by both parties will be over the airline passengers that have been hospitalized for the influenza virus. In the next section, we will illustrate the operation of approximate matching algorithms used in their original forms. Data from this example will be used to do this.

B. Phonetic Algorithms

A phonetic algorithm is an algorithm to match words based on their pronunciation. Phonetic algorithms have been broadly used in the past for record matching performed on names. The main feature of the phonetic algorithms is their fault tolerance against typographical errors. For illustration purposes, we will use Soundex [9] in this paper. However, our methodology can be easily applied to other phonetic algorithms.

The operation of Soundex is quite straightforward: for each word to be encoded certain rules of grouping similar sounds are applied. The result is a four character hash that represents the pronunciation of the word. This hash consists of a capital letter followed by three digits. For example for the word "Cooper", its Soundex code is C160.

C. Distance-Based Methods

Distance-based methods employ functions that map a pair of strings to a real number [36] Levenshtein distance [3] is the best known representative of distance functions. It measures the minimum number of operations required (insert, delete, replace) to transform one string to another. Here, two strings are said to match if their distance is less than d operations, $d > 0$. For example, consider the word "*ryan*" from Table 1, its distance from "*rayn*" of Table 2 will be 2, since two modifications (two replacements) are required to transform one word to another.

D. Bloom Filters

A Bloom Filter [32] is a method for representing a set $A = \alpha_1, \alpha_2, \ldots \alpha_x$ of x elements (also called keys) to support membership queries. The idea is to allocate a vector v of y bits, initially all set to 0, and then choose z independent hash functions, $h_1, h_2, \ldots, h_z$, each with range $1, \ldots, y$. For each element α_i in A, the bits at positions $h_1(a_i), h_2(a_i), \ldots, h_z(a_i)$, in v are set to 1. (A particular bit might be set to 1 multiple times). Given a query for b, we check the bits at positions $h_1(b), h_2(b), \ldots, h_z(b)$. If any of them is 0, then certainly b is not in the set A. Otherwise, we conjecture that b is in the set although there is a certain probability that we are wrong. This is called a "false positive".

A Bloom Filter extension is the Counting Filter proposed by Fan et al. [34]. In this construct, a counter is used consisting of three or more bits, instead of a single bit per position. Thus, the number of elements inserted in the filter may be explicitly computed. The operation of Bloom Filters is based on the hash function used to encode items to be inserted. This function should be

both secure and provide a uniformly distributed hash. For this purpose, we selected the MD5 hashing algorithm [37] due to its broadly studied properties and as it is considered relatively fast.

IV. THE PRIVATE RECORD LINKAGE PROTOCOL

In this section, we will put the pieces together and delve into our proposed protocol. We formally define the following notation to be more precise with our description. A and B are the two data sources used in our setup that will also be referred to as parties or dataholders. Each of them holds data within relations $R_A(A_1, \ldots, A_k)$ and $R_B(B_1, \ldots, B_l)$ respectively. We assume that even though the participating sites do not share a common schema, matching is still performed on a bare minimum of overlapping fields that exist in both sources A and B.

Thus, we will define two additional relation types. First, the *Blocking Schemas* that for A and B will respectively reflect relations R_A^{BL} (A_1^{BL}, …, A_n^{BL}) and R_B^{BL} (B_1^{BL}, …, B_n^{BL}). Each of these relations contains n fields with $n \leq \min(k, l)$ that will be referred to as *Blocking Fields* or *Blocking Keys*. These fields are used in the *Blocking Component* of our framework.

Second, the *Matching Schemas* that for A and B are represented by R_A^M (A_1^M, …, A_m^M) and R_B^M (B_1^M, …, B_m^M), respectively. Each of these relations contains m fields with $m \leq \min(k, l)$, which hence will be termed *Matching Fields or Matching Keys*. These fields are used in the *Matching Component* of our protocol. For either of the two parties, these datasets are related as follows: $R^{BL} \subseteq R$ and $R^M \subseteq R$.

A. Formal Problem Definition

Considering the two data sources A and B, the privacy-preserving record linkage problem boils down to performing record matching between R_A and R_B with an additional constraint. At the end, source A will only know the subset of B's data that match A's data. Equivalently, B will only acquire the subset of A's data that match its own data. Notice here that all the other parameters of the record linkage problem addressed remain as in the classical case. This means that our proposed methodology can equally well deal with non-unique identifiers, as well as data of low quality.

B. Framework Operation Overview

We will now provide a generic view of our protocol before we introduce the reader to the specific components of our solution, as illustrated in Fig. 1. First, the two matching parties A and B will prepare their data to be used in both Private Blocking and Private Matching components. Then A delivers its prepared data to B. B now merges his own blocking data with A's. The secure blocking component is now initiated. After the completion of blocking, the private matching component begins its operation. The overall matching sequence of our framework consists of several blocking and matching passes. Therefore, after secure matching concludes, a new blocking pass begins. When the entire process is complete, B holds the matching results and delivers a copy back to A for further processing.

Our protocol consists of two components. First, the *Private Blocking Component* aims to reduce the candidate pairs between the records to be matched. This is achieved by converting each of the blocking keys to its phonetic code equivalent.

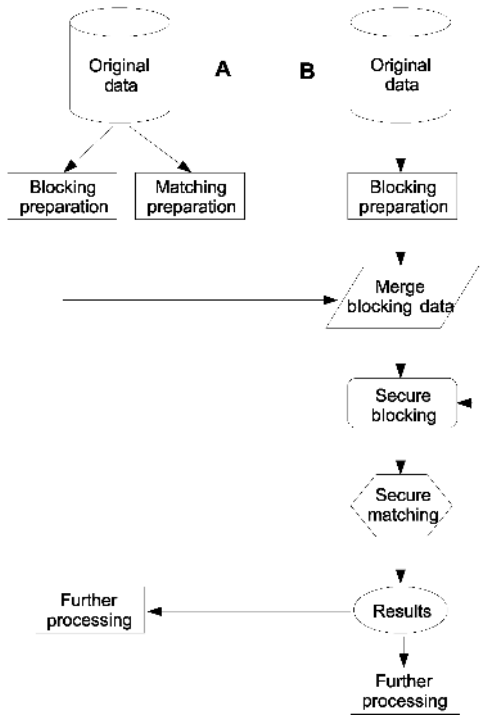

Fig. 1. Protocol overview.

Phonetic codes are used as a first approximate matching indicator, since they feature some tolerance against typological errors. We inject a random number of records consisting of fake phonetic codes to increase fuzziness, i.e., combinations of phonetic codes not found within the ones created by the actual dataset. These codes are then encrypted using a secure hash function. Identical ciphers that represent phonetic codes, are grouped together. Each group indicates which records should be further examined in more detail for matching.

Detailed approximate private matching by means of the Private Matching Component is performed only between records grouped together by the Blocking Component. Consider the main matching phase, sensitive matching data will be hidden by Bloom Filters. Party A will use a Bloom Filter to replace sensitive data for each field used for matching. For each blocking field B will create candidate pairs. Afterwards, the candidate pairs will be examined for matching using relations R_A^M and R_B^M by Bloom Filter based Levenshtein distance algorithm, resulting in a partial set of matching rows. This set is added to the set of total matching rows. This procedure is executed for each of the blocking fields. For the resulting set B_{match} of matched rows, B will append its corresponding non-sensitive information to each Bloom Filter and deliver B_{match} back to A. Finally, A will merge these data with its own.

C. Private Blocking Component

The aim of the Blocking Component is to make the entire

matching procedure more time efficient by reducing the candidate record pairs, while simultaneously maintaining privacy. We employ three levels of security to achieve this: Phonetic encoding, random noise generation, implemented by fake code addition and secure hash encryption.

1) Preparation: The entire process can be outlined as follows. One or more phonetic codes are generated at the respective party for each of the fields in R_A^{BL} and R_B^{BL}, resulting in the datasets $R_{A.Phonetic}^{BL}$ and $R_{B.Phonetic}^{BL}$. Both parties proceed to the same encodings for the same fields. The two new datasets with the phonetic codes may consist of more fields than the initial datasets with the original text fields. This is due to the fact that each blocking key may be encoded with more than one phonetic algorithm.

We follow this process to take advantage of the fault tolerance that phonetic codes exhibit, rendering them appropriate for approximate matching. We use more than one phonetic algorithms for a specific field, since each phonetic algorithm performs better in a different context. Now, a number of fake records comprising phonetic codes are injected into the phonetically encoded datasets. The number and the data of the fakes differ for each of the matching parties. We employ this strategy to artificially increase fuzziness with the phonetics dataset. That is, this approach aims to increase the entropy of the phonetically encoded dataset and as a result, elevate its security.

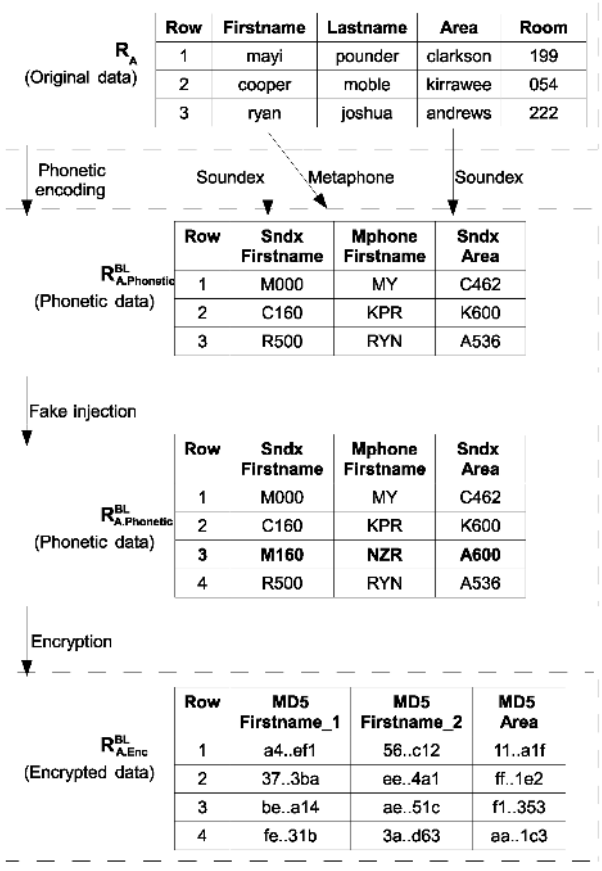

Fig. 2. Blocking preparation at A.

Next, each field belonging to $R_{A.Phonetic}^{BL}$ and $R_{B.Phonetic}^{BL}$ is encrypted by the respective party using the same secure hash function, resulting in relations $R_{A.Enc}^{BL}$ and $R_{B.Enc}^{BL}$ respectively. These relations consist of the same number of fields with the relations holding the phonetic codes. Encryption is performed to both ensure confidentiality during transmission and that during the blocking phase the actual data will not be inferred from their phonetic equivalents, since a single blocking field may have many phonetic transformations featuring high entropy.

We will clarify our approach based on the example data in Tables 1 and 2. Fig. 2 illustrates the preparation for the private blocking step in party A. In our example, we will use fields (*Name*, *Area*) for blocking. For *Name*, we will use Soundex and Metaphone for phonetic encoding, while for *Area* we will only use Soundex.

First, all data within the blocking fields will be transformed to their phonetic equivalents. At this point, it is useful to point out the importance of using different phonetic algorithms over a single blocking field. Considering, the *Name* field in both of the datasets we can see that there is a misspelling regarding the name '*ryan*'. It is spelled correctly in the third row of Table 1, while there is a misspelling in the first row of Table 2. The phonetic encodings of these two versions using Metaphone are 'RYN' and 'RN' while Soundex produces 'R500' for both versions. It is obvious that using a single phonetic algorithm, Metaphone in this case may lead to missing matches. Next, fakes are injected and finally, all phonetic codes are encrypted using a secure hash function. We selected MD5 for this example. The same operations take place at party *B*. The same set of fields is encoded using the same phonetic algorithms, fakes are injected and then all records are encrypted using the same secure hash function.

2) Private Blocking: After both parties complete this procedure, party *A* transmits the dataset consisting of relations $R_{A.Enc}^{BL}$ and R_A^M to party B. Now, for each of the blocking fields a multi pass approach is followed. As suggested in [38], datasets $R_{A.Enc}^{BL}$ and $R_{B.Enc}^{BL}$ are concatenated into a single dataset R^{BL}. This dataset is sorted over the first blocking field that is scanned and sets consisting of identical ciphers are formed. Records within each resulting set are examined with each other within the Private Matching Component. The occurring matching records set is added to the total matching records set that is initially empty. This procedure is repeated for all blocking fields, so that after each pass on a different blocking key, new records are added to the total matching records set.

D. Private Matching Component

The Private Matching Component is responsible for offering elaborate and accurate approximate private matching. It utilizes a secure version of the Levenshtein Distance algorithm that we have developed to achieve this. To the best of our knowledge, distance based matching algorithms share some common characteristics to operate properly. First, the position of each character within a string has to be known. Second, the size of both of the matching strings has to be known to evaluate the various distance metrics. These two parameters play an important role in our implementation. Considering the example scenario pre-

sented previously in section 3, *A* and *B* will agree to merge their data using fields *Name, Lastname* and *Area* that will comprise the matching relations R_A^M and R_B^M.

1) Preparation: We have developed the following methodology to be able to use such characteristics. For each field A_i^M each of its characters is extracted. Each of the characters is appended with a number indicating its position within the string resulting in tokens, consisting of a letter and a number indicating the position of this character within the original string. Next, every token is encoded within a Counting Bloom Filter. We prefer a Counting Bloom Filter, to have exact knowledge of the number of elements inserted. This is achieved by summing the bits of the filter that are set. This number is equal to the string length of the original field A_i^M.

Algorithm 1. Secure edit distance

```
Require: String s, Counting Bloom Filter b
 1: Declare integer d[0.. |s|, 0..|b|]
 2: for i = 0 to |s| do
 3:     d[i, 0] ← i
 4: end for
 5: for j = 0 to |b| do
 6:     d[0, j] ← j
 7: end for
 8: for j = 1 to |b| do
 9:     for i = 1 to |s| do
10:         k ← APPEND(s[i], i)
11:         if k ∈ b then
12:             d[i, j] ← d[i − 1, j − 1]
13:         else
14:             d[i, j] ← min(d[i − 1, j] + 1, d[i, j − 1] + 1, d[i − 1,
                j − 1] + 1)
15:         end if
16:     end for
17: end for
18: return d[|b|, |s|]
```

To be clearer regarding this part of our approach, consider the first row of Table 1 where data from party A are stored. Applying our methodology to word 'mayi', we get four elements: {m1, a2, y3, i4}. Each of these elements will be inserted into the Counting Bloom Filter. After insertion, the sum of bits set by the filter will reflect the number of elements inserted, which in this example equals 4.

A separate filter is used for each field in each row of the matching dataset. After all fields are converted, the matching dataset is transformed to a dataset consisting of Counting Bloom Filters, one for each field of each row. This dataset, which will be referred to as A_{Bloom}^M together with $R_{A\_Enc}^{BL}$, is transferred to party *B*. It is worth mentioning that this procedure is followed by party *A*, no matter which distance metric will be used for matching evaluation.

2) Private Matching: Each of the blocking passes occurring within the Private Blocking component results in sets of candidate matching records that according to our protocol will be compared using some distance metric. We will assume the Levenshtein distance metric, modified for our purposes, to better illustrate our approach. Algorithm 1 displays the modified version to be used with Bloom Filters. The changes to the original algorithm are very limited. Specifically, in line 1, a Counting Bloom Filter is used, instead of providing a second string as input. Next, lines 10 and 11 describe that to each character of the known string, its position within the string is appended, and this token is examined for membership against the filter, instead of comparing characters at specific positions as in the original algorithm.

An example will help clarify our approach. Let us consider the case that a blocking pass over the Name field has brought near records 1 from *A* and 2 from *B*, as illustrated in Tables 1 and 2. When applying Algorithm 1, *s* = '*maya*' and *b* equals to the bloom filter holding elements {m1, a2, y3, i4}. Each character of s is extracted and its position is appended to it. Then, the resulting token is checked against the filter: 'm1', 'a2', 'y3' are found in the filter, while i4 is not. The result of the algorithm is that the Levenshtein distance equals 1.

V. EVALUATION

In this section, we will provide detailed analysis regarding operations taking place at both the Blocking and the Matching Component. The evaluation is made in terms of efficiency and complexity and in terms of protocol security.

A. Efficiency and Complexity

It could be argued that the blocking approach could be used as the main matching procedure, omitting the matching component. However, phonetic codes do not offer detailed matching, leading to increased number of mismatches, having simultaneously increased sensitivity to specific alterations. This renders them unsuitable for detailed matching evaluation.

The use of blocking is essential for the performance of the matching procedure, since it reduces the matching space. Sorting each field of R^{BL} requires $O(nlogn)$ and scanning $O(n)$ operations, reducing the candidate pairs significantly. Comparing all by all matching fields would require $O(n^2)$ comparisons. The decreased complexity of our approach allows applying the blocking passes more than once with different blocking keys.

B. Privacy Analysis

We will present an analysis focusing on two aspects, the information gained by each of the data holders and the information gained by a possible eavesdropper over the transmission channel, to evaluate the privacy offered to the integrated data by our protocol.

1) Privacy during Transmission: Private data belonging either to the matching or to the blocking datasets are encoded using a secure hash function. Therefore, if an eavesdropper gained the linkage file, the difficulty to infer information stored in the Bloom Filters is equivalent to the difficulty of breaking the secure hash function. Moreover, the attacker should be aware of the key used in the hash function. Therefore, the attacker should

pose a brute force attack to identify the hashing key used and the type of matching algorithm used, since all data are broken into tokens depending on the agreed matching technique.

2) Privacy at the Dataholders: We have assumed that both of the data holders follow an Honest But Curious (HBC) behavior [39], trying to infer knowledge by any available information. We will evaluate the privacy of our approach using the metric of *Information Gain*. Information Gain is closely related to *Entropy*. The amount of information in a message is formally measured by the entropy of the message [40]. The entropy is a function of the probability distribution over the set of all possible messages. The entropy of a discrete random variable X is defined as:

$$H(X) = -\sum_{x \in \chi} p(x) \log_2 p(x) \quad (1)$$

Practically speaking, the measure of entropy provides a measure of a set's predictability. Low entropy of X means low uncertainty and as a result, high predictability of X's values. Similarly, the conditional entropy of a discrete random variable Y given the value of the discrete random variable X, $H(Y|X)$ is defined as:

$$H(Y|X) = -\sum_{x \in \chi} p(x)\, H(Y|X = x) \quad (2)$$

Empirically, the notion of conditional entropy aims at quantifying the amount of uncertainty in predicting the value of discrete random variable Y given X.

Information Gain between the discrete random variables Y, X is defined as

$$I(Y;X) = H(Y) - H(Y|X) \quad (3)$$

Information Gain is a metric to assess the difficulty of inferring the original text (Y), knowing its enciphered version (X), or that is, how the knowledge of X's value can reduce the uncertainty of inferring Y. Lower Information Gain means that it is difficult to infer the original text from the cipher. In Section VI, we compare the incurred Information Gain in both the Blocking and the Matching Components to the performance of a rival private record matching algorithm.

VI. EXPERIMENTS

In this section, we provide experiments that prove the effectiveness, the security and the high performance of the proposed framework. We will evaluate the performance of our method in terms of time execution and matching accuracy and compare the results to the performance of the original Levenshtein Distance [3] method and the Bloom Bigram method introduced in [35].

A. Methodology

In our experiments, we will evaluate four methodologies to compare them. First, we consider the original Levenshtein method for reference purposes only. The secure Levenshtein Distance method used in the Private Matching Component of our framework will be the second to evaluate. Of course, we will examine the behavior of the fully deployed framework we propose, consisting of the Private Blocking Component and the Private Matching Component. This is displayed in the graphs as '*Sec. Levenshtein + Sec. Blocking*'. Finally, we will use the Bloom Bigram algorithm as a rival to our framework comparing its security, performance and matching accuracy.

We have used in these experiments the bare minimum of the Blocking Component i.e., without injecting any phonetic codes to be fair. In this way, we will be able to assess the minimum security offered by the Blocking Component, without affecting matching accuracy. The fact that the injection of fake phonetic codes does not affect matching accuracy is indicated in [20].

B. Data Sets Used

We employed two different data sources to conduct our experiments. We used synthetic data generated by the FEBRL [14] data generator for our performance and accuracy experiments and real world data deriving from the DataFerrett [41] utility for our security evaluation. We used FEBRL to take advantage of its ability to generate random datasets, but also due to its feature to create duplicates of these datasets with a variety of error types. The types of alterations created in duplicate datasets range from simple character deletions, replacements or insertions, to field transpositions and nickname transformations (e.g. "Vanessa", "Nessi").

Particularly, we created five different datasets (A-E), as illustrated in Table 3. All these datasets have identical schemas but hold different data. For the duplicates dataset, we required that errors will follow a uniform distribution, with up to 1 error per matching field. The database created by FEBRL has the following schema: *Firstname, Lastname ,Street Number, Address 1, Address 2, Suburb, Postcode, State, Date of Birth, Age, Phone Number, and Socsecid*. We will be using the set {*Firstname, Lastname, Area, Address 1*} as blocking fields and the set {*Firstname, Lastname, Area*} as matching fields. The reason for using DataFerrett was to be able to assess real world data distributions to extract safe conclusions regarding the security of our approach. Specifically, we used data derived from the *Historical Census Data from IPUMS 1920 General Sample* forming dataset F, as illustrated in Table 3. From the schema of this dataset, we used the following field set: *Firstname, Lastname, and Address*.

C. Setup

We used MD5 to construct all the methods requiring Bloom

Table 3. Dataset and experiments

Dataset	Original records	Duplicate records	Source
	Performance and accuracy		
A	10,000	1,000	FEBRL
B	10,000	2,500	FEBRL
C	10,000	5,000	FEBRL
D	10,000	7,500	FEBRL
E	10,000	10,000	FEBRL
	Security evaluation		
F	1,050,634	-	DataFerett

Filters. Each of the Filters consisted of 900 bits, and four different hash functions. An Intel Core2Duo T5550 was used featuring 2GBs of physical memory for all of the experiments. Data was stored in a MySQL 5 database. All algorithms were materialized in Java 6.

D. Privacy

This set of experiments aims at evaluating the security offered by the privacy-preserving methods in our test bed. As such, the original Levenshtein Distance algorithm will not participate in this metric. Moreover, we will separately examine the Private Blocking component and the Private Matching component to be more detailed with our analysis. Specifically, regarding the Private Blocking component, we will present results ranging from one to three blocking keys. These are represented in Table 5 and in Fig. 3 in rows 1 to 3 having respectively Method Ids *A* to *C*.

The methodology followed for this set of experiments is as follows. We measure privacy in terms of Information Gain, as explained previously. We used dataset *F* to perform our evaluation. Specifically, the resulting figures concern calculations over the 'Lastname' field of dataset *F*. We initially encrypted the 'Lastname' field of the dataset using each of the three methods following to evaluate the Information Gain metric for each case. We also used blocking sets {*Lastname, Firstname*} and {*Lastname, Firstname, Address*} for the Private Blocking component. All calculations necessary for our security analysis were performed using SQL queries, as in the one displayed in Table 4.

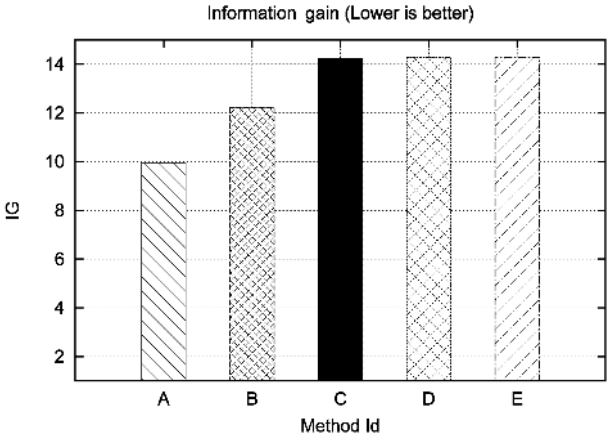

Fig. 3. Information gain (IG) comparison.

Table 4. Sample query for entropy evaluation

```
SELECT
SUM((Occurences/Totals)*log2(Totals/Occurences))
AS Spec_Entropy, Sndx_Lastname
FROM Joint_Soundex_Tables
GROUP BY Sndx_Lastname
ORDER BY Spec_Entropy DESC
```

It has to be clarified that in the case of the Soundex method, the results refer to the plain text version of Soundex. This is due to the fact that during blocking, information is revealed in the form of phonetic codes. Party *B* retrieves phonetic codes, even for records that will not be matched by the Private Matching Component due to the nature of blocking itself. Therefore, it is important to calculate the impact posed to security by our approach.

The results of our experiments indicate, as expected, that the secure Levenshtein Distance approach (illustrated by *D*) is almost equally secure with Bloom Bigrams (illustrated by *E*), since, for the same dataset, the amount of information inferred in both cases is practically equal. This is due to the fact that both approaches use the same medium for encryption, i.e., Bloom Filters.

However, the most interesting result regards the security performance of the Blocking Component. It is evident from Fig. 3 that in this case, statistically speaking, it is much more difficult to infer the original text from the Soundex code than both the Bloom Bigram and secure Levenshtein approaches, even in the case of using two blocking fields. Finally, using three blocking fields proves to be as secure as using the Bloom Bigram Field.

The reason for this is twofold. First, data undergoing the Soundex transformation suffer some information loss, due to the algorithm's nature. Second, many strings may be mapped to the same Soundex code. Conversely, Bloom Filter based approaches result in a much greater number of ciphers of the original text. All these results are detailed in Table 5.

Additionally, someone might argue that if party B had a plaintext list with the original names, it would be able to infer information combining the plaintext with the phonetic code. However, this is not exactly the case. Observing the last column of Table 5, we can see that the information gain, even when using three blocking keys (which in our case is a candidate key for our schema) has lower information gain compared to Secure Levenshtein and Bloom Bigrams.

This result seems awkward at first sight but it is totally explicable considering the properties of phonetic encoding algorithms, since phonetic codes do not feature one to one mapping.

Table 5. Entropy and information gain

Method	Method	No ciphers	H (lastname; cipher)	Information gain
A	Private blocking component, 1 blocking field	3688	4.3310	9.9576
B	Private blocking component, 2 blocking field	805620	2.0773	12.2113
C	Private blocking component, 3 blocking field	707239	0.0579	14.2307
D	Secure Levenshtein	115948	0.0119	14.2768
E	Bloom Bigrams	117865	0.0005	14.2881

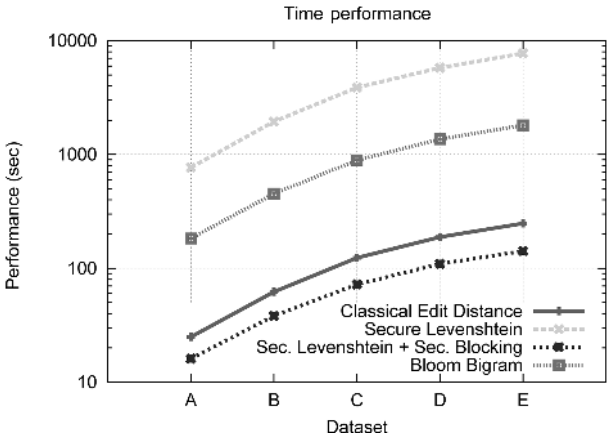

Fig. 4. Time performance comparison.

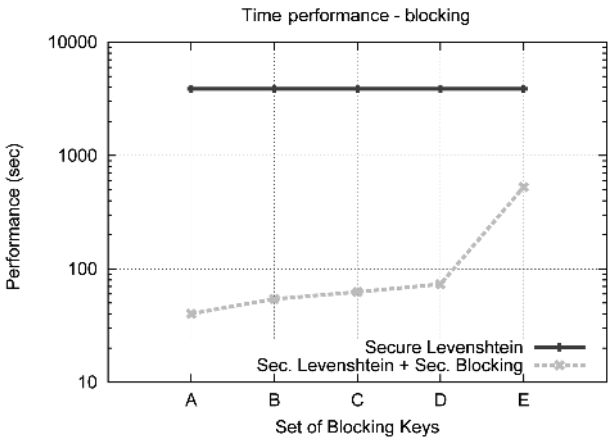

Fig. 5. Blocking keys time performance.

Consider the following case. Someone has a Soundex code and a plaintext that may produce this phonetic code. Even in this case, it is not 100% sure that the given phonetic code refers to the plaintext. For example, both "Smith" and "Sandy" share the same Soundex code, S530.

It is evident that using more blocking fields increases the probability of compromising privacy, but not significantly. Thus, even in the case of three blocking fields, the probability of breaking privacy is marginally lower compared to methods D and E.

E. Execution Time

This set of experiments aims at assessing the performance of our complete framework in terms of execution time and comparing it to both the classical Levenshtein Distance and Bloom Bigrams. We also employ the Private Matching component alone to illustrate the necessity of the blocking step. Fig. 4 compares, using a logarithmic scale, the performance of all the aforementioned methods. As can be seen, our private record linkage framework outperforms both classical Levenshtein and Bloom Bigrams, despite the overhead induced by the cryptographic algorithm used. This clarifies the benefit posed by the Blocking Component. It compensates for the encryption burden, and further improves execution performance.

F. Matching Accuracy

The aim of this set of experiments is twofold. First, we wish to assess the correctness of the secure Levenshtein Distance algorithm we propose. Second, we aim to evaluate the matching accuracy of our private matching framework compared to the original Levenshtein Distance algorithm and Bloom Bigram approach.

We will examine the matching performance of the framework we propose from two aspects. The first regards the behavior of the algorithms for a given matching threshold. That is for distance based algorithms, Edit Distance of 3 and for Bloom Bigrams Dice Coefficient of 0.3 (Fig. 5).

At this point, we consider it necessary to state an interesting detail. Selecting a appropriate matching threshold for the Bigram based approach was tricky, since the Dice Coefficient metric, as presented in [35], depends on the size of the Bloom Filter. As such, this method required some time to properly calibrate the threshold, given our Bloom Filter setup. Conversely, the method we propose is based on the widely used Levenshtein Distance metric that is much easier to calibrate. However, we will not delve into more detail, since this issue goes beyond the scope of this paper.

We employ the *Precision*, *Recall,* and *F-Score* metrics, widely used by the information retrieval community, to evaluate our results. *Precision* is defined as the number of relevant records retrieved by a search divided by the number of records retrieved by that search, and *Recall* is defined as the number of relevant records retrieved by a search divided by the number of existing relevant records (that should have been retrieved). *F-Score* is the harmonic mean of precision recall and defined as

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Regarding the first aspect of our results, we can assess that our private record linkage framework outperforms classical Levenshtein Distance in terms of Precision, as shown in Fig. 6a, while it features slightly lower Recall, as shown in Fig. 6b. Higher Precision occurs due to the Blocking Component, since only records with similar phonetic encoding are examined for matching. Conversely, lower Recall is a side effect of the sensitivity of Soundex to alterations of consonants in the blocking fields. Thus, a totally different Soundex code is generated, leading to lower chances of matching. The fact that our suggested method exhibits similar overall performance to the original version is illustrated by the F-Score metric in Fig. 6c. We also observe that the secure Levenshtein Distance algorithm we propose behaves identically to the original version, a fact that proves its correctness. Moreover, we deduce that the differences in Precision and Recall of our framework from the original Levenshtein Distance are due to the blocking approach we used.

It is evident that our method outperforms Bloom Bigrams in all of precision (Fig. 6a), Recall (Fig. 6b), and F-score (Fig. 6c) metrics. This is mainly due to the fact that our approach without blocking, features identical performance to the classical Levenshtein Distance algorithm, in terms of matching accuracy. Not

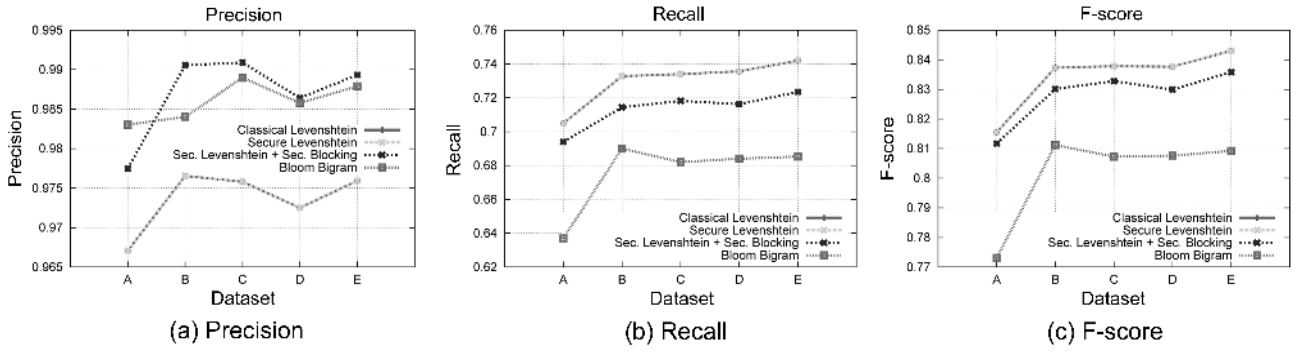

Fig. 6. Results for fixed thresholds.

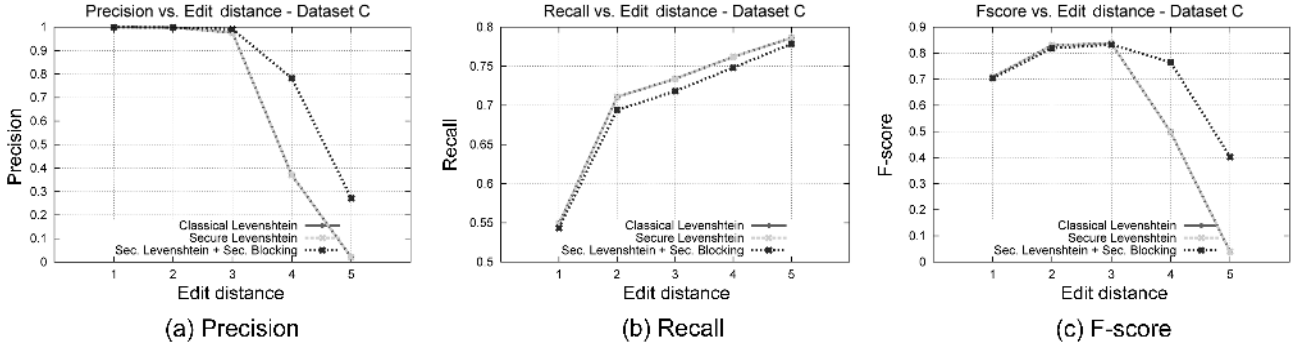

Fig. 7. Precision, recall, Fscore vs. edit distance.

even the use of the Blocking Component degrades the matching performance. All these results lead us to the conclusion that our proposed method is superior to Bloom Bigrams approach in all cases.

Having established the superior security, accuracy and time performance of our private record linkage framework against its rival, it is now time to explore the details of its behavior. Therefore, in the next set of experiments we have not used the Bloom Bigrams method.

In this set of experiments, we will assess the matching accuracy of our performance, experimenting with a variety of Levenshtein Distance thresholds. We have chosen to use dataset C to reduce the space of possible combinations. Examining our results in terms of Edit Distance, we can see in Fig. 7b that again the proposed framework outperforms the original version of Levenshtein Distance, while it closely follows its Recall performance (Fig. 7b). The stability of the suggested method is more evident in Fig. 7c, where we can see that for lower values of Edit Distance, our method behaves similarly to the original Levenshtein algorithm, while for bigger values it significantly outperforms it. Again, this is due to the use of the Blocking Component, where the phonetic algorithm filters most of the unwanted candidate matches, leading to a reduced matching space. Moreover, this set of experiments assures once more the correctness of our approach, since the results between the classical Levenshtein Distance and the secure version we propose are again identical, irrespective of the edit distance. This leads us to the safe conclusion that this approach can be used instead of the classical Levenshtein Edit Distance algorithm, when privacy is required.

G. Blocking Performance

The Blocking Component is an essential part of our protocol, therefore we study separately how its behavior is affected by the blocking keys used. We have used five different sets of blocking keys, as they are listed in Table 6, for this set of experiments. Fig. 5 displays the results concerning the effect on execution time of the various key configurations, while Fig. 8 illustrate the results concerning Precision, Recall and F-Score, for each of the key combinations used. We used dataset C for this set of experiments and considered an edit distance equal to 3. The results are compared to the secure Levenshtein algorithm, which we proved earlier performs identically to its original version, in terms of matching accuracy. It is evident that increasing the number of blocking keys, the quality of results increases, while, as expected, the protocol needs more time to conclude execution. This prop-

Table 6. Blocking key sets used

Set	Blocking keys
A	Name
B	Name, Lastname
C	Name, Lastname, Suburb
D	Name, Lastname, Suburb, Address 1
E	Name, Lastname, Suburb, Address 1, State

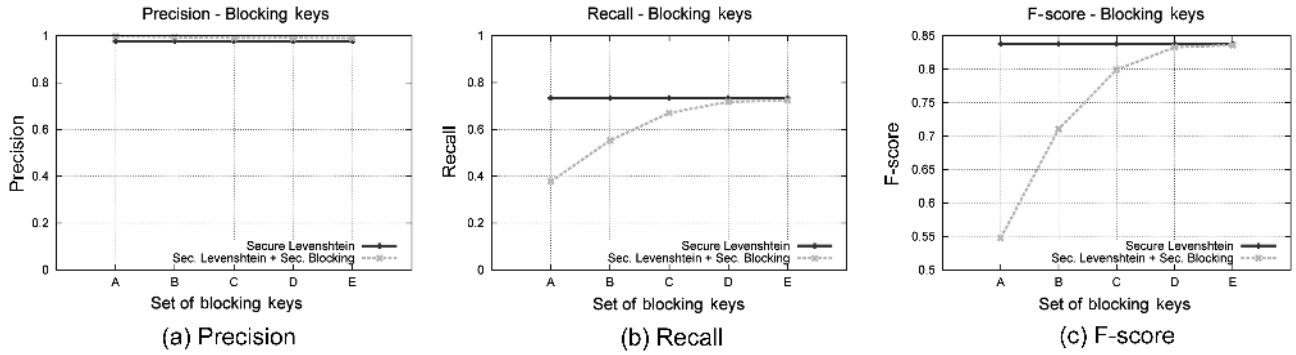

Fig. 8. Precision, recall, F-score vs. blocking keys.

erty allows the application of a variety of strategies during the matching procedure between matching accuracy and time efficiency.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel approach to perform approximate private matching based on the well-known Levenshtein Distance algorithm. Our experiments prove the correctness of our method; it outperforms the original algorithm in terms of execution time, while assuring privacy, and offers almost the same matching performance, which in some cases is superior to the original algorithm.

An important feature of our protocol is that we manage to offer privacy-preserving blocking. Specifically, the Private Blocking component offers the ability to tradeoff matching accuracy and speed, without sacrificing privacy. Depending on the application, we can calibrate the Private Blocking component accordingly.

Our future research directions focus on improving performance in terms of matching accuracy, to overcome the limitations posed by the phonetic algorithms and to explore ways of selecting the blocking keys, in such a way that low execution times are maintained, while matching performance is achieved. Moreover, we are interested in exploring ways to evenly distribute computational burden among matching parties. Finally, we would be interested to modify our protocol to involve more than two matching parties in the computational process.

REFERENCES

1. R. Baxter, P. Christen, and T. Churches, "A comparison of fast blocking methods for record linkage," *Proceedings of the ACM SIGKDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, Washington, DC, 2003, pp. 25-27.
2. The European Union, "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," *Official Journal of the European Union*, vol. L281, pp. 31-50, Nov. 1995.
3. V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707-710, 1966.
4. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1-16, 2007.
5. L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava, "Using q-grams in a dbms for approximate string processing," *IEEE Data Engineering Bulletin*, vol. 24, no. 4, pp. 28-34, 2001.
6. M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414-420, 1989.
7. W. E. Winkler, The State of Record Linkage and Current Research Problems, Washington, DC: Statistical Research Division, US Bureau of the Census, 1999.
8. W. E. Winkler, Overview of Record Linkage and Current Research Directions, Washington, DC: Statistical Research Division, US Census Bureau, 2006.
9. M. K. Odell and R. C. Russell, US Patent Number 1261167, 1918.
10. L. Philips, "Hanging on the metaphone," *Computer Language*, vol. 7, no. 12, pp. 39-43, Dec. 1990.
11. L. E. Gill, "OX-LINK: the Oxford medical record linkage system," *Record Linkage Techniques--1997: Proceedings of an International Workshop and Exposition,* Arlington, VA, 1997, pp.15-33.
12. R. L. Taft, Name Search Techniques. Special Report / New York State Identification and Intelligence System, Albany, NY: Bureau of Systems Development, 1970.
13. M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid, "TAILOR: a record linkage tool box," *Proceedings of the 18th International Conference on Data Engineering*, San Jose, CA, 2002.
14. P. Christen, "Febrl--an open source data cleaning, deduplication and record linkage system with a graphical user interface," *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, 2008, pp. 1065-1068.
15. W. W. Cohen, "The WHIRL approach to integration: an overview," *Proceedings of the AAAI-98 Workshop on AI and Information Integration*, Madison, WI, 1998, pp. 26-27.
16. C. Clifton, M. Kantarcioglu, A. Doan, G. Schadow, J. Vaidya, A. Elmagarmid, and D. Suciu, "Privacy-preserving data integration and sharing," *Proceedings of the 9th Workshop on Research Issues in Data Mining and Knowledge Discovery, In Conjunc-*

tion with ACM SIGMOD International Conference on Management of Data, Paris, France, 2004, pp. 19-26.
17. T. Churches and P. Christen, "Blind data linkage using n-gram similarity comparisons," *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, 2004, pp. 121-126.
18. D. Sankoff and J. B. Kruskal, Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Stanford, CA: Center for the Study of Language and Information, 1999.
19. V. S. Verykios, A. Karakasidis, and V. K. Mitrogiannis, "Privacy preserving record linkage approaches," *International Journal of Data Mining, Modelling and Management*, vol. 1, no. 2, pp. 206-221, 2009.
20. A. Karakasidis and V. S. Verykios, "Privacy preserving record linkage using phonetic codes," *The 4th Balkan Conference in Informatics*, Thessalonikei, Greece, 2009, pp. 101-106.
21. S. Trepetin, "Privacy-preserving string comparisons in record linkage systems: a review," *Information Security Journal: A Global Perspective*, vol. 17, no. 5-6, pp. 253-266, Dec. 2008.
22. D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *IEEE Symposium on Security and Privacy*, Berkeley, CA, 2000, pp. 44-55.
23. W. Du and M. J. Atallah, "Protocols for secure remote database access with approximate matching," *Proceedings of the 7th ACM Conference on Computer and Communications, and the First Workshop on Security and Privacy in E-Commerce*, Athens, Greece, 2000.
24. E. Van Eycken, K. Haustermans, F. Buntinx, A. Ceuppens, J. Weyler, E. Wauters, H. Van Oyen, M. De Schaever, D. Van den Berge, and M. Haelterman, "Evaluation of the encryption procedure and record linkage in the Belgian National Cancer Registry," *Archives of Public Health*, vol. 58, no. 6, pp. 281-294, 2000.
25. A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco, "A hybrid approach to private record linkage," *Proceedings of the 24th International Conference on Data Engineering*, Cancun, Mexico, 2008, pp. 496-505.
26. M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid, "Privacy preserving schema and data matching," *ACM SIGMOD International Conference on Management of Data*, Beijing, China, 2007, pp. 653-664.
27. M. Kantarcioglu, W. Jiang, and B. Malin, "A privacy-preserving framework for integrating person-specific databases," *Privacy in Statistical Databases UNESCO Chair in Data Privacy International Conference, PSD 2008*, Istanbul, 2008, pp. 24-26.
28. S. S. Bhowmick, L. Gruenwald, M. Iwaihara, and S. Chatvichienchai, "PRIVATE-IYE: a framework for privacy preserving data integration," *Proceedings of the 22nd International Conference on Data Engineering*, Atlanta, GA, 2006, pp. 91-91.
29. R. Hall and S. E. Fienberg, "Privacy-preserving record linkage," *Proceedings of the International Conference on Privacy in Statistical Databases*, Corfu, Greece, 2010, pp. 269-283.
30. A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino, "Private record matching using differential privacy," *Proceedings of the 13th International Conference on Extending Database Technology: Advances in Database Technology*, Lausanne, Switzerland, 2010, pp. 123-134.
31. M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, Washington, DC, 2003, pp. 39-44.
32. B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422-426, 1970.
33. G. Koloniari and E. Pitoura, "Distributed structural relaxation of XPath queries," *Proceedings of the 25th International Conference on Data Engineering*, Shanghai, China, 2009, pp. 529-540.
34. L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281-293, Jun. 2000.
35. R. Schnell, T. Bachteler, and J. Reiher, "Privacy-preserving record linkage using Bloom filters," *BMC Medical Informatics and Decision Making*, vol. 9, no. 1, p. 41, Aug. 2009.
36. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," *Proceedings of the IJCAI 2003 Workshop on Information Integration on the Web*, Acapulco, Mexico, 2003, pp. 73-78.
37. R. Rivest, "The MD5 message-digest algorithm," http://www.ietf.org/rfc/rfc1321.txt?number=1321.
38. M. A. Hernandez and S. J. Stolfo, "Real-world data is dirty: data cleansing and the merge/purge problem," *Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 9-37, 1998.
39. O. Goldreich, Foundations of Cryptography, Vol 2: Basic Applications, New York, NY: Cambridge University Press, 2004.
40. C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379-423, Jul. 1948.
41. US Census Bureau, "DataFerrett," http://dataferrett.census.gov/.

Alexandros Karakasidis

Alexandros Karakasidis received his Computer Science Degree from the Department of Computer Science of the University of Ioannina in 2002. He received his MSc from the same department in 2005. Currently, he is a PhD candidate in the Department of Computer and Communication Engineering at the University of Thessaly, in Volos, Greece. His main research interests include privacy preserving record linkage, privacy, security and anonymity in advanced database systems, data mining and data quality. He has published papers in refereed journals, international conferences and workshops. He has served as a reviewer for International journals and conferences.

Vassilios S. Verykios

Vassilios S. Verykios received the Diploma in Computer Engineering from the University of Patras in Greece, in 1992 and the MS and the PhD degrees from Purdue University in 1997, and 1999 respectively. He has been an associate professor in the School of Science and Technology, at the Hellenic Open University in Patras, Greece, since January 2011. His main research interests include data management, privacy, security and anonymity, data mining, data reconciliation, and privacy preserving record linkage. He has published over 70 papers in major referred journals and in the proceedings of international conferences and workshops, and he has coauthored a monograph on Association Rule Hiding for Data Mining by Springer.