Privacy Preserving Probabilistic Record Linkage Using Locality Sensitive Hashes

Ibrahim Lazrig[1], Toan Ong[2], Indrajit Ray[1(✉)], Indrakshi Ray[1], and Michael Kahn[2]

[1] Department of Computer Science, Colorado State University, Fort Collins, USA
{lazrig,indrajit,iray}@cs.colostate.edu
[2] Anschutz Medical Campus, University of Colorado, Denver, USA
{Toan.Ong,Michael.Kahn}@ucdenver.edu

Abstract. As part of increased efforts to provide precision medicine to patients, large clinical research networks (CRNs) are building regional and national collections of electronic health records (EHRs) and patient-reported outcomes (PROs). To protect patient privacy, each data contributor to the CRN (for example, a health-care provider) uses anonymizing and encryption technology before publishing the data. An important problem in such CRNs involves linking records of the same patient across multiple source databases. Unfortunately, in practice, the records to be matched often contain typographic errors and inconsistencies arising out of formatting and pronunciation incompatibilities, as well as incomplete information. When encryption is applied on these records, similarity search for record linkage is rendered impossible. The central idea behind our work is to create characterizing signatures for the linkage of attributes of each record using minhashes and locality sensitive hash functions before encrypting those attributes. Then, using a privacy preserving record linkage protocol we perform probabilistic matching based on Jaccard similarity measure. We have developed a proof-of-concept for this protocol and we show some experimental results based on synthetic, but realistic, data.

1 Introduction

The problem of privacy preserving record linkage arises in many data sharing applications that limit the access to certain information on a need-to-know basis. A typical example of this problem is that of record linkage in large clinical research networks. As part of increased efforts to provide precision medicine to patients, large clinical research networks (CRNs) are building regional and national collections of electronic health records (EHRs) and patient-reported

This work was partially supported by a grant from Colorado State University, by NIST under contract 70NANB158264, and by NSF under grants IIP1540041 and CNS1619641. Any opinions, findings, recommendations, or conclusions reported in this work are those of the author(s) and do not necessarily reflect the views of the NIST or the NSF.

S. Ranise and V. Swarup (Eds.): DBSec 2016, LNCS 9766, pp. 61–76, 2016.
DOI: 10.1007/978-3-319-41483-6_5

outcomes (PROs). To protect patient privacy, each data contributor to the CRN (for example, a health-care provider) uses anonymizing and encryption technology before publishing the data. A medical researcher (data subscriber) on the other hand requests medical data from different data contributors and needs to aggregate the records of different patients. The challenging scenario occurs when many competitor data publishers are willing to anonymously and securely share some information with a subscriber, but do not want to share the encryption keys in order to protect their clients' privacy. The researcher is interested in finding linked (matched) patient records across the datasets and to be able to retrieve updates on some previously queried patients as well. In addition, the subscriber is not only interested in finding deterministically matched records only, but also records that potentially matches with high probability.

If the data is encrypted with a shared (or same) key, the matching process could be performed deterministically [11]. However, in a typical real-world setting, where the data publishers operate independently of each other, there is no guarantee that the same datum is consistent across different publishers. This occurs owing to factors such as, different structures (FName versus First Name), semantics (whooping cough vs. pertussis), typographic and formatting errors (John vs. Jhon), or the way the data is captured, entered and maintained by each entity. As a result, deterministic matching becomes difficult or ineffective. If the data is encrypted with different keys (as would typically be done by independent data publishers that do not trust each other), finding matches between the records are even more challenging. For privacy preserving record matching to be effective under such circumstances, we need to consider as many of the discrepancies in the data as possible that can potentially occur during acquisition (input) such as spelling mistakes, formatting errors, abbreviations, punctuations, pronunciations, users background, or different underlying assumptions about the structure of the data, and attempt to determine how similar the encrypted data are.

In this work, we build upon our previous secure record matching protocol [11], that performs a deterministic record matching using the services of a *semi-trusted* broker, in order to construct a *probabilistic* record matching protocol. It takes into account the different types of discrepancies mentioned above. Probabilistic in the context of this work means that the matching is performed based on the likelihood of similarity with certain probability. We call the set of attributes used to perform the record matching, the *linkage attributes*. (Note that in the literature the term "quasi-identifier" is often used for the combination of attributes used for linkage. We do not use the term "quasi-identifier" because it explicitly excludes attributes that are themselves unique identifiers. In our case, the linkage attributes can be key attributes too.) In order to hide the identities of the matched entities from all of the parties, the matching process is performed on encrypted data under different keys. Any two records of different data sets are said to be matched if they belong to the same individual/entity. In our demonstration, we use personal identification information as the linkage attributes (e.g. names. SSN, email address, DOB, Driving License number, etc.) for finding candidates for record matching. However, our protocol is easily generalizable to any other attributes that are of interest.

1.1 Related Work

The record linkage problem is defined as the process of finding records belonging to the same entity, across two or more data sets [6]. It has been widely studied in the context of linking records stored on different databases so as to identify duplicate records that appear to be different. Approximate comparison of string values and distance measures allow researchers to deal with data discrepancies and errors that causes such apparent differences in duplicate records [2,12,13,15]. In this work, we rely on Jaccard similarity measure that is calculated based on the common sub-strings of bi-grams between the strings to be compared. The min-hash technique, which has been proven to be a good approximation to the Jaccard similarity [2,12], is used to construct a locality-sensitive hash function that speeds up the calculation of the Jaccard similarity [12].

Sharing individual data, such as medical or financial records, among multiple parties is a two edge sword that has its research benefits and privacy concerns for both the parties (who might be competitive) and the subjects (individuals) whose data is to be shared. Because privacy is a big concern, many research has been conducted to address privacy issues in the record linkage of shared data. The research resulted in the development of a number of privacy preserving record linkage protocols. Some of the protocols allow the parties to directly communicate and perform the matching process, and some utilize the service of a third party to accomplish the task of record matching. Our work belongs to the second category, where the third party blindly performs the matching process.

Some research developed protocols targeted at private set intersection [3,7,9] and secure multi-party computation (SMC) [1,4,5,8] to solve the record linkage problem. However these techniques do not fit directly in our scenario because either they require a shared key or the parties involved learns the result of the computation. Further more, some of these techniques do not scale very well with the number of parties and the size of the data sets, and incur large computation or communication overhead [19].

In [10], a framework for privacy-preserving approximate record linkage is proposed. The framework is based on a combination of secure blocking and secure matching. The secure blocking is based on phonetic algorithms statistically enhanced to improve security, and the secure matching is performed using a private approach of the Levenshtein Distance algorithm. The main advantage of blocking is that it results in a huge decrease of record comparisons.

A three-party protocol, proposed in [17], relies on identical encryption at the sources and uses Bloom filters to perform similarity search. The linkage center performs probabilistic record linkage with encrypted personally identifiable information and plain non-sensitive variables. To guarantee similar quality and format of variables and identical encryption procedure at each site, the linkage center generates semi-automated pre-processing and encryption templates based on information obtained via a new method called data masking that hides personal information.

Similarity preserving data transformation approaches like [14,16,18,20] have been also presented. In [16], a protocol that provides privacy at the levels of

both data and schema matching was presented. In this protocol, records are mapped into an Euclidean space using a set of pre-defined reference values and distance function, while preserving the distances between record values. Then the semi-trusted (HBC) third party will compare the records in the metric space in order to decide their matching. The secure schema matching requires global schema, provided by the third party, on which the parties map their own local schemas. In [20], an approach based on the work in [16], that doesn't need a third party was presented. A complex plain is created then an adjustable width slab is moved within the complex plain to compute likely matched pairs. In [14], public reference tables are used to encode names as the distance to all reference points. Then a third party will estimate the distance between pairs based on their encoding.

In [18], a Bloom filters with a set of cryptographic hash functions are used to encode the attributes of the data records. The encoded records are then compared via a set-based similarity measure.

Most of these protocols where either theoretically or experimentally proven to be efficient. However most of these works do not fit directly into our scenario [1,3,7,9,19] Because The data source parties participate in the matching process will know the matching results, or they work for deterministic match [3,7], require a shared key [17], or rely on expensive SMC operations that do not scale very well with the size of the data [1,4,5,8].

2 Problem Formulation, Definitions and Background

We abstract the problem as that of a group of $n \geq 2$ publishers (data sources) $\{P_1, \cdots, P_n\}$, with n data bases $\{D_1, \cdots, D_n\}$ who would like to share information with another group of $m \geq 1$ subscribers (e.g. researchers) $\{S_1, \cdots, S_m\}$ while protecting the privacy of their clients. At the end of the linkage process, we require that 1) the subscriber(s) gets the linked records identified by some random non-real ids, 2) none of the publishers knows the result of the linkage process, or that a certain client's information exists at other party's records, 3) none of the parties, including the party conducting the linkage process, should know the real identity of any client that it did not know a priori.

Definition 1 - *Linkage Attributes:* *For a database D_i of publisher P_i with a schema $C = (A_1, \cdots, A_w)$ that has a set of w attributes, we define the set of linkage fields $L \subseteq C$ as a subset of $t \leq w$ attributes that uniquely define the personal identity of the database records. We denote the linkage fields set as $L = \{L_1, \cdots, L_t\}$ and the set of records of D_i as $R = \{R_1, \cdots, R_v\}$. We refer to the value of the linkage field L_j of the record R_k by $R_k \cdot L_j$.*

Definition 2 - *Linkage Parameters:* *We associate with the linkage fields L sets of linkage parameters K, St, W, such that for each linkage field $L_i \in L$, we define some linkage parameters, $K_i \in K$ that represents the number of hash functions used for similarity calculations of the values of that field, similarity threshold $0 \leq St_i \leq 1$ which defines the minimum similarity threshold to consider*

the values of the linkage field as a match, and Linkage Field weight $w_i \in W : \sum_{1 \leq j \leq t} w_j = 1$ *that represents the importance of that field in the matching decision.*

Definition 3 - *Field Match Score:* *Given two values* $R_j \cdot L_i, R_k \cdot L_i$ *of linkage field* L_i *of two records* R_j, R_k*, similarity function Sim, field similarity threshold* St_i*, and field weight* w_i*, a field match score* FM_i *is defined as:*

$$FM_i(R_j \cdot L_i, R_k \cdot L_i) = \begin{cases} 0 & if\ Sim(R_j \cdot L_i, R_k \cdot L_i) < St_i \\ w_i \cdot Sim(R_j \cdot L_i, R_k \cdot L_i) & otherwise \end{cases}$$

Definition 4 - *Record Match Decision:* *Given a similarity threshold* TR *and a set of Field Match scores* $\{FM_i : 1 \leq i \leq t\}$ *of two records* R_j, R_k*, the record match decision* RM *of* R_j, R_k *is defined as:*

$$RM(R_j, R_k) = \begin{cases} Match & if \sum_{1 \leq i \leq t} FM_i \geq TR \\ Non - Match & otherwise \end{cases}$$

2.1 Adversary Model

This work derives from the deterministic matching protocol of our previous work [11] that relies on a *semi-trusted* broker. Data publishers are considered competitors who do not want to share data with each other. Each publisher tries to determine information about the clients of competitor publisher, or at a minimum, tries to determine if any one of its clients is also a client of its competitor. However, publishers are also honest in the execution of the protocol steps and are willing to share information with subscribers *privately*, that is, without revealing real identities, and *securely*, that is, without any leakage of information to other data publishers, and without revealing the publisher's identity to the subscribers.

A data subscriber, on the other hand, needs to determine if any information that came from different publishers belong to the same individual so they could be grouped together as such and treated accordingly. For example, if a researcher is looking for the side effects of a new drug used for skin treatment on patients who has kidney problems, then he has to match patients from the Dermatology and Nephrology departments to find patients under these conditions. We need to allow such grouping at the subscriber side.

Further more, the subscriber is allowed to issue *retrospective* queries regarding some individual client, for example, update queries regarding the progress of treatment of certain patients. Subscribers (researchers) are considered curious in the sense they will try to determine the real identities of the individuals. Some information about individual identities might be leaked from their non-identification information (i.e. eye color, age, weight, etc.) using statistical inference techniques. This is a separate problem that needs to be addressed with anonymization (i.e. k-anonymity) or other sanitization methods, and is not considered in this work.

The broker is honest in the sense that it will not collude with any of the parties, but is curious and not trusted to keep a secret, secret. The broker will

work as a mediator between the data publishers and the subscribers by honestly performing the following tasks:

- Hide the source of information (publishers and clients' identities) from the subscribers.
- Blindly determine record linkages among the encrypted publishers' records and assign alternate random identifiers to the linked records before sharing them with the subscribers. The broker will just know the linkages between two encrypted records without knowing the real identifiers.

2.2 The Secure Deterministic Matching Protocol

The current work builds on our previous work [11] where we introduced the deterministic matching protocol using a semi-trusted broker. To facilitate understanding of the current protocol we briefly describe the previous work. Our previous protocol has three phases, the *setup phase*, the *encryption of query results* phase, and the *secure matching phase*:

- *Setup phase*: It is executed only once. In this phase the publishers collaboratively and securely create a set of *encryption key converters*, one for each publishers, using El-Gamal homomorphic encryption and the broker's public key pk. The inputs to this phase are each publisher's initial temporary encrypted key converter $t_{i\rightarrow i} = \mathsf{Enc}_{\mathsf{pk}}(r_i^{-1})$, and the publisher's secret key $\mathsf{sk_i}$, where r_i is an initial random secret of this publisher d_i. At the end of this phase, each publisher has its initial key converter value ($t_{i\rightarrow final}$) securely processed (using homomorphic operations under the broker's public key) by all other publishers. Then publisher d_i sends its final encrypted key converter value to the broker, and saves a copy of it for future updates in case new publishers join the system or to refresh its secret key. Then, For each $t_{i\rightarrow final}$, the broker extracts the key conversion δ_i using its private key sk such that:

$$\delta_i = \mathsf{Dec}_{\mathsf{sk}}(t_{i\rightarrow final}) = r_i^{-1} \prod_{\substack{j=1 \\ j\neq i}}^{N} sk_j$$

- *Encryption of query results phase*:
  Each publisher has the publicly known ElGamal EC parameters, i.e., the curve parameters $E(\mathbb{F}_q)$ and the point on the curve P of prime order n. The public/private key pair will be $(r_i \cdot \mathsf{sk_i} \cdot P, r_i \cdot \mathsf{sk_i})$ and both of the keys are kept secret. Each publisher encrypts the identification part of its data set id, which in our multiplicative scheme needs to be a scalar. We denote by $\mathsf{E}(.)$ the encryption of ElGamal based on EC.
  The publisher first hashes the identifying part of every record in its data set using a universal hash function H. Then uses its secret key multiplied by the corresponding random value, $(r_i \cdot \mathsf{sk_i})$, to encrypt the resulting hash. That is, the encryption of any identifier id will be:

$$\mathsf{E}_{(r_i \cdot \mathsf{sk_i})}(H(id)) = H(id) \cdot r_i \cdot \mathsf{sk_i} \cdot P$$

Finally, the publisher substitutes the real identifying part, id, by $\mathsf{E}_{(r_i \cdot \mathsf{sk}_i)}(H(id))$ for all records in its data set before being sent to the broker. Publishers can avoid having the broker store the key converter $(\delta_i)_{i \in [N]}$. For this purpose, each publisher encrypts the identifiers of the query results with a new random value r_i, updates the key converter $t_{i \rightarrow final}$, then sends these results to the broker accompanied with the new key converter. This solution adds negligible communication overhead, but ensures a zero-key stored on the broker side.

– *Secure matching phase*:
The broker receives the encrypted identifiers under different keys from different publishers. The broker's job is to merge similar clients' records from different publishers such that they will map to the same newly generated identifier. It uses the δ_i values to convert any identifier id encrypted by publisher d_i under its secret key $(r_i \cdot \mathsf{sk}_i)$, to a value encrypted under a different unknown secret key Δ, i.e., $\mathsf{E}_{\Delta}(H(id))$. The key $\Delta = \prod_{i=1}^{N} \mathsf{sk}_i$ is resulting from the product of all the secret keys of all publishers, and cannot be computed by the broker who posses only the key converters of publishers. In order to perform the secure record matching, the broker re-encrypts the encrypted identifying parts of the records coming from the publisher d_i using the corresponding key converter δ_i as:

$$\mathsf{E}_{\delta_i}\left(\mathsf{E}_{(r_i \cdot \mathsf{sk}_i)}(H(id))\right) = \mathsf{E}_{\Delta}(H(id))$$

3 The Secure Probabilistic Matching Protocol

In order to allow the parties to perform secure similarity match based on Jaccard measure, we improve our previous deterministic protocol to perform the match probabilistically with a predefined threshold. We improve the string set representation of the matching (linkage) attributes, then create signatures based on Locality Sensitive Hash scheme explained below. To prevent any correlation analysis, the signatures are lexicographically sorted and encrypted using the publishers' secret keys.

3.1 Minhash Functions and Similarity

To calculate the similarity of two strings A and B, we use the Jaccard Similarity measure. First the two strings are converted to a set representation S_A, S_B respectively (e.g. using bi-grams). Then the Jaccard similarity, $Sim(A, B)$, is calculated on S_A and S_B as:

$$Sim(A, B) = \frac{\mid S_A \cap S_B \mid}{\mid S_A \cup S_B \mid}$$

If $Sim(A, B)$ is close to 1, then A and B are very similar. Let Ω be the set of all possible values of the set representation of the strings (e.g. all possible bi-grams of the alphabet), then MinHash (also called min-wise hash) is computed

by applying a random permutation $\pi : \Omega \rightarrow \Omega$ on any given set S and selecting the minimum value.

$$MinHash_{\pi}(S) = min\{\pi(S)\}$$

It was proven by A. Broder [2] that the probability of the minhashes of any two sets S_A and S_B being the same is equal to the Jaccard similarity of the two sets; that is:

$$Pr(MinHash_{\pi}(S_A) = MinHash_{\pi}(S_B)) = \frac{\mid S_A \cap S_B \mid}{\mid S_A \cup S_B \mid} = Sim(A, B)$$

Thus, we can estimate the Jacard similarity of the two strings A and B by choosing a set of n (e.g. $n = 100$) independent random permutations, apply them to each string set representation to get a set of n minhashes (we also use the term signatures) of those strings, and compute how many corresponding minhashes of the two sets are equal. It is possible to simulate the n random permutations by n independent random hash functions (e.g. selecting a hash function and use n different random seeds).

3.2 Locality Sensitive Hashing – LSH

Locality sensitive hashing or LSH is a well known technique in the information retrieval community, that is used for determining similar items by hashing them in such a way that their hash values will collide with high probability. Let St_1, St_2 be a similarity metric (e.g. Jaccard similarity) and P_1, P_2 be two probabilities, then a family of functions $\mathcal{H}$ (e.g. minhashes) is called the (St_1, St_2, P_1, P_2)-sensitive LSH if for every $h \in \mathcal{H}$ and any x, y:

1. if $Sim(x, y) \geq St_1$, then the probability that $h(x) = h(y)$ is at least P_1.
2. if $Sim(x, y) \leq St_2$, then the probability that $h(x) = h(y)$ is at most P_2.

From above, a family of K minhash functions with a similarity threshold St_0 is a $(St_0, 1 - St_0, St_0, 1 - St_0)$-sensitive LSH for Jaccard similarity. For a better accuracy of the similarity estimation it is preferred to have St_1 and St_2 as close as possible and P_1 and P_2 as far as possible. We construct a $(St_0, 1 - St_0, \gamma, \beta)$-sensitive LSH family from a family of K minhash functions by grouping the minhashes (while keeping the order) into b groups of size r each, such that $K = b \cdot r$, and then hash each group to get b new signatures. Hence, $\gamma = 1 - (1 - St_0^r)^b$ and $\beta = (1 - St_0^r)^b$, where the probability that the new b signatures of x and y agree in at least one signature is γ.

We can further reduce the number of false positives by requiring that the b-signatures of x and y agree on at least m signatures (e.g. $m = 2$), then the probability of the b signatures to agree in at least m signatures will be:

$$\sum_{m \leqslant i \leqslant b} \binom{b}{i} St_0^{r \cdot i} (1 - St_0^r)^{b-i}$$

However, experiments show that $m = 1$ is sufficient.

3.3 Choosing LSH Parameters

For a similarity threshold St_0, and K minhash functions we set the values of b (number of groups/signatures) and r (number of minhash values in each group) such that for any two strings x,y that have $Sim(x, y) \geq St_0$, the probability of getting at least m out of b of their corresponding signatures to match is greater than 0.5. This is done as follows:

$m \simeq b \cdot St_0^r$ and $b = K/r$ so $r = K \cdot St_0^r/m$ or $r = ln(r \cdot m/K)/ln(St_0)$, solve for r then use $b = \lfloor K/r \rfloor$ to calculate b. All the values K,b, and r must be integers. The similarity threshold could also be approximated based on r and b as $St_0 \simeq (1/b)^{1/r}$.

3.4 String Set Representation

In order to apply the LSH technique to get the signatures of the strings, we first need to convert the strings to sets of Q-grams (e.g. Bi-grams, for Q=2). In our implementation we extend the bi-gram set representation of our strings in such a way that it accounts for missing and flipped character positions. We add bi-grams constructed from tri-grams with the middle character being deleted. For example, the *the extended set representation* of the string **abcde** is constructed as follows:

- create the set of bi-grams B as $B = \{\_a, ab, bc, cd, de, e\_\}$
- create the set of Tri-grams T as $T = \{\_ab, abc, bcd, cde, de\_\}$
- From T create the set of extensions X as $X = \{\_b, ac, bd, ce, d\_\}$
- The Extended set E will be $E = B \cup X$

With this extension the possibility of matching strings with flipped character increases; for example the two strings **John** and **Jhon** will be considered a match with .75 Jaccard similarity using this technique. While the same strings without this extension will have .25 Jaccard similarity. We represent dates as strings and append some characters before and after the month, day and year components to make the bi-grams generated form those components look different in case they have the same digits.

3.5 Creation of LSH Signatures

During the setup phase, all data sources (publishers) agree on the set of size t linkage fields $L = \{L_1, \cdots, L_t\}$ and their corresponding similarity computation parameters, which are K_i hash functions (or, it could be one hash function with K_i different seeds), and similarity threshold St_i for each linkage field $L_i \in L$. We would like to emphasize here that those linkage fields not necessarily a single database attribute, they might be concatenations of certain attributes, as it is the case in many record matching works. Then each party calculates b_i (number of groups/signatures) and r_i (number of minhash values in each group) for each record linkage field L_i as discussed in Sect. 3.3 above. The steps for signature creation are as follows: We denote by $Sig_j^{(i)}$ the set of signatures of the column

i of record j, and $ESig_j^{(i)}$ is its corresponding encrypted signatures. Each party A with records set $R_A = \{R_1, \cdots, R_n\}$ having linkage fields $L = \{L_1, \cdots, L_t\}$, creates the linkage field signatures for each record $R_j \in R_A$, $1 \leq j \leq n$ as follows:

- For each linkage field value $R_j.L_i \in R_j$ create its corresponding Encrypted signature $ESig_j^{(i)}$ as follows:
  1. Convert the value $R_j.L_i \in R_j$ to an extended bi-grams set E
  2. Apply K_i hash functions to the set E and get the K_i minhash values $minh = \{min(h_f(E)) : 1 \leq f \leq K_i\}$
  3. Group the set $minh$ into b_i groups of r_i items each, and without changing their order, i.e. $G_x = \{minh[(x-1)*r_i+1] \| \cdots \| minh[x*r_i]\}, 1 \leq x \leq b_i$. Then, using a universal hash function H, hash each group into one value getting the set of b_i signatures of the value $R_j.L_i$ as: $Sig_j^{(i)} = \{H(G_g) : 1 \leq g \leq b_i\}$
  4. The encrypted signatures of the value $R_j.L_i$ will be: $ESig_j^{(i)} = \{E_{sk_{A,L_i}}(Sig_j^{(i)}[g] || g || L_i) : 1 \leq g \leq b_i\}$, where sk_{A,L_i} is the secret key of party A for linkage attribute L_i. This secret key could be the same for all attributes.
- sort $ESig_j^{(i)}$ values in lexicographical order and append them as new field to the record.

3.6 Record Matching Phase

Each data source (publisher) will send the encrypted signatures of the linkage fields (columns) (as separate or combined) table(s) along with random record identifiers of its data set to the broker. To determine if record $A.R_k$ matches the record $B.R_j$, the broker executes the following steps:

- For each set of encrypted signatures of each linkage column ($L_i \in L$) of both records i.e. $A.R_k.ESig_k^{(i)}$, and $B.R_j.ESig_j^{(i)}$, apply the key-conversion as follows:
  Use the key converter of A (δ_A values) to convert the signatures of linkage fields $A.R_k.ESig_k^{(i)}$ of the record $A.R_k$ encrypted by publisher A under its secret key sk_A, to signatures encrypted under a different secret key Δ, i.e., $\mathsf{E}_\Delta(Sig_k^{(i)})$ for every linkage field L_i. The key $\Delta = \prod_{p=1}^{N} \mathsf{sk}_p$ is resulting from the product of all the secret keys of all parties (publishers). That is,

$$\mathsf{E}_{\delta_A}\left(ESig_k^{(i)}\right) = \mathsf{E}_{\delta_A}\left(\mathsf{E}_{\mathsf{sk}_A}(Sig_k^{(i)})\right) = \mathsf{E}_\Delta(Sig_k^{(i)})$$

  The same applies to B, that is,

$$\mathsf{E}_{\delta_B}\left(ESig_j^{(i)}\right) = \mathsf{E}_{\delta_B}\left(\mathsf{E}_{\mathsf{sk}_B}(Sig_j^{(i)})\right) = \mathsf{E}_\Delta(Sig_j^{(i)})$$

- Compare the resulting signature sets encrypted under the same key Δ, $B.\mathsf{E}_\Delta(Sig_j^{(i)})$ and $A.\mathsf{E}_\Delta(Sig_k^{(i)})$ to find the number of equal signatures es_i and

update the results table for this linkage field L_i. If the number of equal signatures is zero (0), then the similarity between these two values of this linkage field is less than the pre-set similarity threshold St_i, and hence the field match score $FM_i(R_j \cdot L_i, R_k \cdot L_i)$ will be zero. Otherwise, calculate the Estimated similarity of the two records for this linkage field L_i as $S_i \simeq (es_i/b_i)^{1/r_i}$, then calculate the field match score based on its weight w_i using the Eq. 3 shown in Sect. 2. $FM_i(R_j \cdot L_i, R_k \cdot L_i) = w_i \cdot S_i$

– To make a record match decision $RM(R_j, R_k)$ and declare the records R_j and R_k as a match or not, first compute the overall record match score using the field match scores computed in the previous step.

$$RecScore = \sum_{1 \leq i \leq t} FM_i$$

Then compare the record match score with the pre-set record match threshold TR as shown in Eq. 4 in Sect. 2. If a match is found, save the matched record identifiers as a matched pair in the results table.

At the end, the broker will have an association (mapping) between the record random identifiers of the publishers based on the scores computed from the matched signatures of the corresponding fields. For example, suppose we have two publishers A, and B that send the encrypted signatures of their records as tables with the following schema:

Publisher A: $A.col_1SigTable(ARecId,\ Col_1Sigs),\ A.col_2SigTable(ARecId,\ Col_2Sigs),\ \cdots,\ A.col_tSigTable(ARecId,\ Col_tSigs)$
Publisher B: $B.col_1SigTable(BRecId,\ Col_2Sigs),\ B.col_2SigTable(BRecId,\ Col_2Sigs), \cdots,\ B.col_tSigTable(BRecId,\ Col_tSigs)$

Then the Broker will find the matched records and create the mappings between A's record Ids and B's record Ids and save them as new table with a schema similar to this: $matchingResultTable(ARecId, BRecid, Score)$.

4 Implementation and Results

To evaluate the performance and the accuracy of this protocol, we implemented it and tested it against some data sets of different sizes and using combinations of different linkage fields. We adopted the following two approaches, (1) using each single attribute as a linkage field, and (2) using concatenation of some attributes as a linkage field. We evaluated the accuracy and performance based on these two approaches.

We define accuracy in terms of *precision* and *recall* based on true positive (TP), false positive (FP), true negative (TN) and false negative(FN). **TP** (True Positive) is the number of originally-matching records that are correctly identified as match. **FP** (False Positive) is the number of originally-non-matching records that are falsely identified as match. **TN** (True Negative) is the number of originally-non-matched records correctly identified as a non-match. Finally, **FN**

(False Negative) is the number of originally-matched records falsely identified as a non-match. Based on these values, we compute the following:

1. **TPR** (True Positive Rate) Or Sensitivity/recall: Fraction of originally matched records that are correctly declared as match: $TPR = \frac{TP}{TP+FN}$
2. **TNR** (True Negative Rate) Or Specificity: Fraction of originally non-matched records that are correctly declared as non-match: $TNR = \frac{TN}{TN+FP}$
3. **PPV** (Positive Predictive Value) Or Precision: Fraction of correctly matched records from all records declared as a match: $PPV = \frac{TP}{TP+FP}$
4. **ACC** (Accuracy): Fraction of records correctly declared as a match and correctly declared as non-match: $ACC = \frac{TP+TN}{TP+TN+FP+FN}$
5. F_1 **score** (F-Measure) is the harmonic mean of precision (PPV) and recall (TPR), calculated as $F_1 score = 2 \times \frac{(PPV \times TPR)}{(PPV+TPR)}$.

4.1 Results of Using Realistic Synthetic Dataset

We use two datasets generated by the *Mocaroo* realistic data generator (http://www.Mocaroo.com) and consisting of 10 K records each, and 6 K of the records are true matches. The record attributes (fields) used in this data were (ID, SSN, FirstName, LastName, email, DOB, Gender, ZipCode). We used two versions of the datasets, one is more corrupted than the other. We opt to use realistic synthetic datasets for many reasons, like known linkage results, shareable dataset for reproducibility (i.e., synthetic data don't require IRB approval), and controlled error/missing values rate. The data contains randomly generated real names, SSNs, addresses, and so other attributes. The corruption percentage is based on the number of modified (removed/inserted/swapped) characters to simulate errors occurred when acquiring the data (input into the database). The greater the data corrupted the more errors it contains.

We ran an extensive set of experiments with different configurations on both versions of the datasets (less corrupted and highly corrupted), and for convenience we will discuss only some of them here. After some experiments using signatures for single attributes, we picked similarity threshold for each attribute (low threshold to allow more permissiveness, and high threshold for strictness in the matching criteria).

1. **Using the Less corrupted Vs. the highly corrupted Datasets:** The more corrupted the data the lower the similarity between the records will be. So the similarity threshold of each linkage field that effect the its signatures creation will effect the matching results as well. For more permissiveness the threshold should be low, and vice versa. However the lower threshold will effect the False positive rate, so it is better to keep false positive as low as possible. For example, the F1-score result of two experiments of low (ssn = 0.85, first = 0.65, last = 0.65, email = 0.75, DateOB = 0.85, zip = 0.85) and high (ssn = 0.85, first = 0.75, last = 0.75, email = 0.75, DateOB = 0.85, zip = 0.85) thresholds settings of seven different sets of combinations of the single attributes, and using the less corrupted data is shown in Fig. 1, where the sets used are:

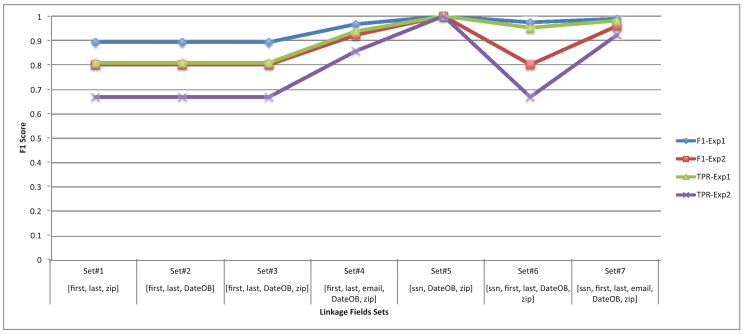

Fig. 1. F1 score using low (Exp1) and high (Exp2) Similarity thresholds for single attributes, and different combinations (Color figure online)

- Set#1: [first, last, zip]
- Set#2: [first, last, DateOB]
- Set#3: [first, last, DateOB, zip]
- Set#4: [first, last, email, DateOB, zip]
- Set#5: [ssn, DateOB, zip]
- Set#6: [ssn, first, last, DateOB, zip]
- Set#7: [ssn, first, last, email, DateOB, zip]

2. **Using Single Attributes Vs. Concatenated Attributes as Linkage Attributes**
   Using logical expressions constructed from combinations of single attributes gives more flexibility to form matching criteria, and allows fine tuning of the matching threshold of each attribute separately. However it incurs more computation and storage overhead and make the signatures more susceptible to frequency attacks on some attributes. On the other hand, using the concatenated values of certain attributes before creating the signatures will limit the matching criteria to the combinations of these concatenations and reduce the flexibility of matching conditions. However such concatenated attributes, if properly constructed, will reduce the computations required in both signature creation and matching process, and limit the frequency attacks.
   In our experiments we used the first technique to evaluate the accuracy of the protocol, and since the signatures are encrypted from the source with different keys, the frequency analysis is only possible if conducted by the broker. If the publishers use new random with each signature, update their key-converter δ and include it with the signatures, they can limit the broker frequency attack to those matched attributes, though will add communication overhead.
   Table 1 shows the results of using four different combinations of the set of attributes. The top part of the table for the combinations using single attributes signatures, each has its similarity threshold, the bottom part of the table for the concatenated attributes and the similarity threshold is set for the whole concatenated attributes of each combination. The combinations are as follows:

- comb1: First Name, Last Name, Date of birth
- comb2: Date of birth, SSN
- comb3: Last name, SSN
- comb4: Three Letters First Name, Three Letters Last Name, Soundex(First Name), Soundex (Last Name), Date of birth, SSN

Any two records matched by any of the above combinations is considered a match. From the results in Table 1, we were able to fine tune the threshold similarity of each attribute, using the single attribute technique, to control the true positives and false positive and get better results. It is much harder to do that in the second technique. The combination (Comb4) where constructed by creating a new attribute (we named "Special1") that consists of the concatenations of the first four parts of comb4 above (i.e. "Three Letters First Name" + "Three Letters Last Name" + "Soundex (First Name)" + "Soundex (Last Name)"), then create the signatures for it.

Table 1. Matching quality results of using single and concatenated combinations of attributes with different similarity thresholds settings

Using signatures of single attributes						
Sim. Threshold for (SSN,First,Last,DOB,Special1)	TP	FP	PPV	ACC	F1	Time (ms)
(0.85, 0.85, 0.85, 0.85, 0.85)	5161	14	0.997	0.915	0.924	384589
(0.85, 0.85, 0.85, 0.95, 0.85)	4916	2	1	0.891	0.901	261302
(0.85, 0.75, 0.75, 0.98, 0.85)	5022	0	1	0.902	0.911	498038
Using signatures of concatenated attributes						
Sim. Threshold for (comb1, comb2, comb3, comb4)	TP	FP	PPV	ACC	F1	Time (ms)
(0.95, 0.95, 0.95, 0.95)	5140	9	0.998	0.913	0.922	3895
(0.98, 0.988, 0.99, 0.99)	4710	0	1	0.871	0.880	2009
(0.98, 0.98, 0.98, 0.95)	4848	0	1	0.885	0.894	2364

Finally, Table 1 also shows the time used in the matching process using both techniques. It is evident how the second technique out performed the first, for the reasons mentioned above. In conclusion, our system achieved good matching quality results with good performance.

5 Conclusion

In this paper, we addressed the problem of privacy preserving probabilistic record linkage across multiple encrypted datasets when the records to be matched contain typographic errors and inconsistencies arising out of formatting and pronunciation incompatibilities, as well as incomplete information. We create characterizing signatures for the linkage of attributes of each record using minhashes and locality sensitive hash functions before encrypting those attributes. Then, using a privacy preserving record linkage protocol we perform probabilistic matching based on Jaccard similarity measure. We show some experimental results based on synthetic, but realistic, data. The results show that our matching protocol is flexible

and achieved good matching quality without too much computation or communication overhead. Our future work will focus on improving the matching quality results by incorporating automation methods for selecting the best matching criteria and properly adjusting the similarity thresholds and weights of the attributes. In terms of security we will work on making the secure matching protocol resilient to collusion by making the broker collude with at least k out of n parties in order to get the encryption key.

References

1. Atallah, M.J., Kerschbaum, F., Du, W.: Secure and private sequence comparisons. In: Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, WPES 2003, pp. 39–44. ACM, Washington, DC (2003)
2. Broder, A.: Identifying and filtering near-duplicate documents. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 1–10. Springer, Heidelberg (2000)
3. De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010)
4. Durham, E., Xue, Y., Kantarcioglu, M., Malin, B.: Quantifying the correctness, computational complexity, and security of privacy-preserving string comparators for record linkage. Inf. Fusion **13**(4), 245–259 (2012)
5. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M.J., Wright, R.N.: Secure multiparty computation of approximations. ACM Trans. Algorithms **2**(3), 435–472 (2006)
6. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. J. Am. Stat. Assoc. **64**(328), 1183–1210 (1969)
7. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
8. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)
9. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: Proceedings of the 19th Annual Network and Distributed System Security Symposium, San Diego, California, February 2012
10. Karakasidis, A., Verykios, V.S.: Secure blocking + secure matching = secure record linkage. J. Comput. Sci. Eng. **5**(3), 223–235 (2011)
11. Lazrig, I., et al.: Privacy preserving record matching using automated semi-trusted broker. In: Samarati, P. (ed.) DBSec 2015. LNCS, vol. 9149, pp. 103–118. Springer, Heidelberg (2015)
12. Leskovec, J., Rajaraman, A., Ullman, J.: Mining of Massive Datasets. Cambridge University Press, Cambridge (2014)
13. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
14. Pang, C., Gu, L., Hansen, D., Maeder, A.: Privacy-preserving fuzzy matching using a public reference table. In: McClean, S., Millard, P., El-Darzi, E., Nugent, C. (eds.) Intelligent Patient Management. SCI, vol. 189, pp. 71–89. Springer, Heidelberg (2009)

15. Porter, E.H., Winkler, W.E., Census, B.O.T., Census, B.O.T.: Approximate String Comparison and Its Effect on an Advanced Record Linkage System. U.S. Bureau of the Census, Research report (1997)
16. Scannapieco, M., Figotin, I., Bertino, E., Elmagarmid, A.K.: Privacy preserving schema and data matching. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007, pp. 653–664. ACM, Beijing, China (2007)
17. Schmidlin, K., Clough-Gorr, K.M., Spoerri, A.: Privacy preserving probabilistic record linkage (P3rl): a novel method for linking existing health-related data and maintaining participant confidentiality. BMC Med. Res. Methodol. **15**(1), 1–10 (2015)
18. Schnell, R., Bachteler, T., Reiher, J.: Privacy-preserving record linkage using bloom filters. BMC Med. Inform. Decis. Mak. **9**(1), 1–11 (2009)
19. Vatsalan, D., Christen, P., Verykios, V.S.: An efficient two-party protocol for approximate matching in private record linkage. In: Proceedings of the Ninth Australasian Data Mining Conference, AusDM 2011, vol. 121, pp. 125–136. Australian Computer Society, Darlinghurst, Australia (2011)
20. Yakout, M., Atallah, M.J., Elmagarmid, A.: Efficient private record linkage. In: Proceedings of the 25th IEEE International Conference on Data Engineering, ICDE 2009, pp. 1283–1286, March 2009