Blocking-Aware Private Record Linkage

Ali Al-Lawati
Penn State / CSE
allawati@cse.psu.edu

Dongwon Lee
Penn State / IST
dongwon@psu.edu

Patrick McDaniel
Penn State / CSE
mcdaniel@cse.psu.edu

ABSTRACT

In this paper, the problem of quickly matching records (i.e., record linkage problem) from two autonomous sources without revealing privacy to the other parties is considered. In particular, our focus is to devise *secure blocking* scheme to improve the performance of record linkage significantly while being secure. Although there have been works on private record linkage, none has considered adopting the blocking framework. Therefore, our proposed blocking-aware private record linkage can perform large-scale record linkage without revealing privacy. Preliminary experimental results showing the potential of the proposal are reported.

1. INTRODUCTION

The task of integrating similar databases populated at separate locations to improve data qualities and enable accurate data analysis is often restricted by heterogeneity in the data. Specifications of how data is represented differ across databases of different parties. For example, "penn state university" can appear as simply "penn state", or as "The Pennsylvania State University". The goal of *record linkage* [16] is to identify similar records with precision, thereby facilitating accurate pattern and data analysis. In record linkage, all data to be matched appears in its original form in the matching process. This is acceptable when the data is of little value or when participants are mutually trusting. However, it is inappropriate to reveal privacy of autonomous sources. The goal of *private record linkage* [7] is, thus, to identify similar records without revealing privacy to others [1]. Applications that demand such a private record linkage include the medical field where patient records must be shared among hospitals and institutions while the identity of the patients are sealed.

Given two data sources, X and Y, the most naive form of record linkage is to perform pair-wise comparison – each record x from X and y from Y are compared one by one, having a quadratic time complexity of $O(|X||Y|)$. Furthermore, each record x and y are typically examined by some distance metrics – if $dist(x, y)$ exceeds some threshold then the pair is "matched". One of the popular techniques to improve the performance of record linkage methods is to use *blocking* – by clustering records into pre-determined blocks so that expensive distance measures are performed to only records within each block. Typical blocking works as follows: Suppose one pre-groups records in Y into Y/b blocks (i.e., each block has b records on average). Then, each record x from X is compared to only records from one block and the matching record is determined. That is, the time complexity is reduced to $O(|X||b|)$. Since $b \ll X$, this blocking in general improves the performance.

As the data size grows and more expensive distance metrics are employed, the importance of blocking increases as well. However, in the case of private record linkage, to our best knowledge, no previous approaches attempt to combine the "blocking" with the "private record linkage." Therefore, in this paper, we study a *blocking-aware private record linkage* protocol and propose several blocking schemes for enhanced performance. A key observation is the preprocessing of records into blocks, using "secure blocking" schemes.

Example 1 (Motivation). To illustrate the benefits of blocking-aware private record linkage, consider two credit card companies that wish to identify fraudulent customer list common to both companies. However, using the regular record linkage is inappropriate because of the private nature of credit card information (i.e., two companies do not want to share their customer-related information with the other party). Assume company A holds 2 million records and company B holds 3 million records. Moreover, suppose an average of 50 records are assigned to each block. In the absence of blocking, every record pair has to be compared, resulting in 2 million $\times$ 3 million = 6 trillion comparisons! However, blocking reduces the number of comparisons to 2 million $\times$ 50 = 100 million. □

2. RELATED WORK

By and large, three categories of previous work are closely related to ours: secure data sharing, record linkage, and private record linkage.

Secure data sharing. Private record linkage applies when the underlying data is of sensitive nature. Such privacy aware protocols are categorized under the emergent field of privacy preserving data mining: a topic of wide recent interest. Most work in the area is either in data perturbation, or secure data sharing. Data perturbation algorithms [17, 14] distort individual fields of a database for pri-

vacy, but preserve the overall structure of a database. This enables the extraction of aggregate data or patterns without disclosing raw data. On the other hand, secure data sharing considers sharing or querying of selected data while securing all other. Unlike secure data sharing, private record linkage applies when data is heterogeneous.

Many protocols for secure data sharing have been described in the literature. Yao et al [26] describe the first two-party sharing protocol; more protocols and cryptographic constructions that support multiparty sharing are presented in [15]. Secure data sharing protocols assume a *semi-honest* or *honest-but-curious* [3] behavior from the participating parties. This means parties follow the protocol without cheating, but may try to find as much information about one another's databases. Other work by Evfimievski et al [8, 9] investigates the secure mining of association rules, evaluation of breeches, and quantification of privacy.

The work of Agrawal et al [1] defines minimal information sharing in the context of secure data sharing protocols as the set of additional categories of information inferred by parties. The protocols they describe, however, are not useful for matching of heterogeneous data.

Record Linkage. Record linkage is a problem that has received wide attention from different communities. The statistical field is interested in probabilistic approaches to evaluating the similarity of records [24, 10, 13]. In the AI field, training data is fed to distance metrics in order to enhance their ability to recover duplicate records [22]. The information retrieval and database fields are mainly concerned with merging heterogeneous databases and optimizing queries using constructions such as inverted indexes.

To improve the performance of record linkage for large datasets, blocking (sampling) schemes are described in the literature. The essence of blocking is similar to indexing; in the database field, indexing increases the performance of query processing by maintaining tables of related records. Parallel to indexing, several distinctions arise in blocking such as manual vs. automatic, or controlled vs. uncontrolled vocabulary [21]. For generality, it is important that blocking is automated and accommodates any vocabulary.

Blocking schemes reduce the set of candidate record pairs for which more expensive distance metrics, such as TFIDF [21] or Jaro [2], are computed. Baxtor et al [2] analyze several different blocking schemes and compare them for accuracy and reduction ratio. An effective sampling algorithm is described in [12, 11] based on records' textual attributes. Experimentation shows that even a simple blocking scheme can result in significant performance gains, with minimal impact on precision.

Private Record Linkage. Much of the work in private record linkage has been pioneered in the medical field. Of mention is the work of Quantin et al [19, 18], inspired by the confidentiality needs of medical information. In them, Quantin et al describe a keyed hashing transformation protocol for representing epidemiological data as per the requirement of European privacy laws for irreversible transformations of such data. Matching of similar data is computed on the transformed set. More recent trends with privacy needs include cooperation among government agencies, selective sharing of intellectual property, and outsourcing [1]. Churches et al [4] analyze security of their protocols as a function of the number of mediating parties. They refer to their techniques as "blindfolded" record linkage.

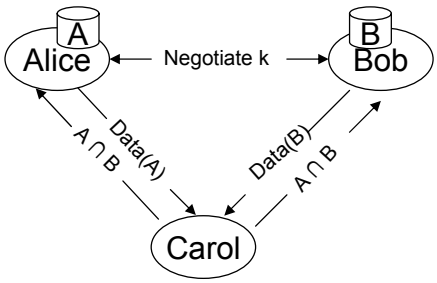


Figure 1: A general third-party matching protocol.

More recent is the work of Ravikumar et al [6] which proposes a secure, stochastic private record linkage protocol that implements all distance metrics where records are representable in a weight vector, such as TFIDF and Soft-TFIDF [5]. Calculation of vectors' similarity relies on a secure intersection algorithm to compare tokens with probability proportional to their weight. Precision asymptotically converges to the true value as the number of record samples increases. Nonetheless, this protocol's use of a secure intersection algorithm translates into expensive computation. Such algorithms rely on commutative hashing based on expensive public key encryptions. Furthermore, the protocol requires multiple occurrences of each record to accurately observe the probabilistic model (i.e., training set). Overall, this work is a valuable contribution that paves the way for two-party private record linkage, however, the constructions upon which it is based lack practical maturity.

3. PRIVATE RECORD LINKAGE

3.1 Overview

Suppose two autonomous parties wish to compute the private record linkage problem on their databases. The task is to design a blocking-compatible private record linkage protocol for enhanced performance. The protocol computes the matching set of records securely, minimizing any information leakage.

We adopt a third party approach to solve this problem. Figure 1 illustrates the communication steps needed. To secure the contents of databases A and B, Alice and Bob must negotiate a secret key and use it to achieve data confidentiality. In this Section, we define our protocol and show how we use *hash signatures* to achieve confidentiality and compactness. Later, we describe several blocking schemes and experimentally verify the performance gains achieved.

3.2 Threat Model and Evaluation

Participants in a private record linkage problem are characterized by *semi-honest*, or *honest-but-curious* behavior [3]. Semi-honest behavior presumes a party will attempt to infer any information possible from the data supplied by other parties. This includes carrying out frequency analysis and known-text attacks of data. We adopt a conservative approach to security based on the premise that an adversary has access to a pool of all known text.

Nonetheless, semi-honest behavior forbids participants from other forms of cheating. Participants do not misrepresent

their inputs by supplying false data (spoofing attack) or intentionally hiding parts of their data (hiding attack) [25]. No form of collusion between any party and a third party occurs. Semi-honest behavior is a common requirement for participants in secure data sharing problems in the literature.

In minimal information sharing, security is determined by the additional categories of information divulged in the process of solving a problem. For example, depending on the distance metric used, the third party is given access to information (e.g. weight information) that facilitates matching of records. However, there exist other important categories of information with more tunable levels of exposure. The following categories of information are considered in our analysis of information leakage:

- Database size (DB_{size}).
- Vocabulary size of a database ($Vocab_{size}$).
- Lengths of database records (Rec_{len}).
- Frequency of all tokens in a database (Tok_{frq}), not to be confused with TFIDF's token frequency (TF).

For each category above, we define three levels of exposure:

- Yes (divulged, revealed, determined, exposed): the category is accurately measured by a curious party.
- **inf**: (1) an upper bound is computed on the whole category, or (2) an accurate measure is calculated on a subset of a category.
- No: the category is not divulged, neither is an upper bound divulged.

Determining a level of exposure on the defined categories is a loose characterization of a protocol's privacy. In Section 4, we employ this characterization to analyze the performance of our protocol with respect to blocking scheme alternatives.

3.3 Protocol

Several participants interact in a private record linkage problem. We first define the participants as follows: There are three semi-honest participants: *Alice*, *Bob*, and *Carol*. Alice holds private database A that contains a set of records with a finite number of fields. Bob holds another database B that contains a similar but separate set of records. Alice and Bob are the collaborating parties who wish to find the common records in their databases A and B. Carol is the third party chosen by Alice and Bob to execute the distance metric.

Figure 1 illustrates a general third party matching protocol. The protocol assumes all communication between parties is carried over a secure channel, e.g., via cryptography. A general overview of the steps of our private record linkage protocol that incorporates blocking is as follows:

1. Alice and Bob negotiate a secret key, k, unknown to Carol.
2. Alice and Bob each use a blocking method to generate blocks and assign records of databases A and B to the blocks. Next, Alice and Bob transmit the blocks to Carol.
3. Carol computes a distance on the reduced set of candidate pairs and forwards the results to Alice and Bob.

Third party requirement is common among private record linkage protocols because techniques used in two party protocols fail to correctly represent and match heterogeneous data. Two party protocols rely on double encryptions of data, the output of which are matched. Double encryptions do not preserve properties of heterogeneous data. In [4], the authors further expand on the number of third parties and analyze security as a function of the number of compromised third parties. In [6], the authors describe a third party free protocol for private record linkage, but it achieves poor performance as we mentioned previously.

Third party protocols use keyed transformations of records, e.g keyed hashing, to prevent security breeches due to curious behavior from the third party. Otherwise, the third party can analyze collaborating parties' databases with similar transformations performed on a pool of all known text (i.e brute force).

3.4 Secure Hashing

We introduce the *hash signature* transformation construct as a compact and secure way to represent TFIDF weight vectors. TFIDF is a proven distance metric based on the tokens (words) in a record. Hash signatures use keyed hashing of individual tokens of a record using a pre-negotiated key k. For brevity, k is omitted but implied in the computation of a hash signature. Before presenting further details, it is useful to first introduce TFIDF.

TFIDF. The TFIDF [21] (Token Frequency / Inverse Document Frequency) distance metric is widely used for matching of similar information. TFIDF is based on the intuition that tokens which appear frequently in a given record should be assigned higher weights in that record (TF weight), while tokens which appear frequently in the database as a whole should be generally assigned low weights (IDF weight). For example, consider a database of universities. TFIDF would assign higher weights to tokens such as "PSU" and "Stanford", and lower weights to tokens such as "university" and "college". Records are matched according to a normalization of the linear combination of the TF and IDF weights of similar tokens. Two records A_x and B_y are considered similar if their TFIDF score exceeds some threshold. The TFIDF score is given by:

$$TFIDF(A_x, B_y) = \frac{weight_x \cdot weight_y}{|weight_x| \times |weight_y|} \quad (1)$$

To compute $weight_x$, for each $w_i \in A_x$,

$$weight(x, w_i) = log(TF_{w_i} + 1) \times log(IDF_i) \quad (2)$$

It follows,

$$weight_x = \bigcup_i weight(x, w_i) \quad (3)$$

Alternatively, Eq 1 can be represented as a dot-product of two vectors V_x and V_y [6].

$$V_x = \frac{weight_x}{|weight_x|} \quad (4)$$

Hence, Alice and Bob can independently compute V_x and V_y. The vector lengths are dependent upon the size of the vocabulary set of the two databases.

Database A		Database B	
id	record	id	record
a1	{'a', 'b'}	b1	{'b'}
a2	{'c'}	b2	{'a', 'b'}

Table 1: Databases A and B

	$F[0]$	$F[1]$	$F[2]$	$F[3]$
$HS(a1)$	weight(a1,'b')	0	0	weight(a1,'a')
$HS(a2)$	0	0	weight(a2,'c')	0
$HS(b1)$	weight(b1,'b')	0	0	0
$HS(b2)$	weight(b2,'b')	0	0	weight(b2,'a')

Table 2: Example hash signatures

TFIDF specifies measuring the TF weight of tokens on a record granularity, but measures the IDF weight, i.e. IDF_i, on a common vocabulary of $A \cup B$. For Alice to correctly represent V_x, token information is required from database B, and vice versa. Note that it may be insecure to divulge the IDF weights of the vocabulary of either database. However, it is conceivable to assume that databases A and B are similar, e.g., if database A consists of citation data, database B will also consist of citation data. Hence, separate IDF measurements appear to be good estimates for a common vocabulary.

Hash Signature. As mentioned above, hash signatures are compact and secure representations of TFIDF weight vectors, namely, V_x and V_y. The size of vectors V_x and V_y is not fixed and is based on the data residing in databases A and B. This results in long vectors and requires insecure pre-agreement on vocabulary ordering. A hash signature transformation of A_x denoted as $HS(A_x)$ is a compact, vocabulary-independent representation of the TFIDF weight vector based on a simple hashing function with a small output t, e.g., 10 bits. The hash function is evaluated on each token, and the hash output is an index to a 2^t floating point array where the weight is stored. As with V_x and V_y, the TFIDF score of a pair of records is a dot-product of the hash signatures.

For example, let $A_x = \{w_1, ..., w_n\}$, F be an array of floating point values initialized to 0, and h^t be a hash function with t bit output.

$$\begin{gathered} F[i_1] = weight(x, w_1) \\ F[i_2] = weight(x, w_2) \\ ... \\ F[i_n] = weight(x, w_n) \\ i_l = h^t(w_l), l \in \{1...n\}^1 \end{gathered}$$

$$HS(A_x) = F \tag{5}$$

Example 2. Assume Alice's database A contains 2 records and Bob's database B also contains 2 records as in Table 1. To illustrate how a hash signature is computed, let the hash function h^t output be $t = 2$. As mentioned previously, the hash function performs keyed hashing with an omitted key k: $h^t('a') = 3$, $h^t('b') = 0$, and $h^t('c') = 2$. Each hash signature is composed of a floating point array of weights, as shown in Table 2. The array position where a weight of a particular token is stored depends on the hash output of the token. For example, weight(a1,'b') is stored in the entry with index 0, because $h^t('b') = 0$. □

[1] *Actually*, $h^t(w_i||k)$ is computed, but k is omitted for brevity. The symbol $'||'$ denotes string concatenation.

3.5 Analysis

Notice that since the hash output size, t, is relatively small, there is a high probability of collisions. This means distinct tokens of a record may map to the same hash signature entry. Likewise, dissimilar tokens of a record pair may map to the same entry. This can result in inaccurate TFIDF scoring. In our implementation, if multiple tokens of a record map to the same entry, only the smallest weight is stored to minimize the effect on TFIDF scoring. Assuming that every collision incident results in incorrect matching:

Lemma 1. *The worst case performance of our protocol using hash signatures is* $\frac{2^t!}{(2^t-n)!\times 2^{nt}}$*, where* n *is the number of different tokens in* $A_x \cup B_y$*.* ■

Proof. Assume a worst case performance such that every collision occurrence results in a mismatch. We want to show the worst case performance of TFIDF using hash signatures is $\frac{2^t!}{(2^t-n)!\times 2^{nt}}$ of the performance of TFIDF. But, given $HS(A_x)$, and $HS(B_y)$, the number of permutations such that two different tokens in the the pair map to the same array entry $= \binom{2^t}{n} \times n!$. By dividing the total number of possibilities, we have $\frac{\binom{2^t}{n}\times n!}{2^{nt}} = \frac{2^t!}{(2^t-n)!\times 2^{nt}}$. (q.e.d)

Hence, for $n \ll 2^t$, the worst case achieves performance above 95% of performance of TFIDF. Note, t is a design parameter that can be adjusted to satisfy $n \ll 2^t$.

A further modification geared at conserving disk space is possible with hash signatures. Since $n \ll 2^t$, hash signatures are sparse weight arrays. It follows that the set of database records is an instance of a sparse matrix. There are many ways to compactly represent sparse matrices, including smaller arrays of positions and corresponding values or a linked list of position/value pairs. However, we consider a technique known as *run-length encoding*, which lists each weight followed by the number of zeros suppressed. Run-length encoding is suitable to our scheme because it is easy to implement and features a small overhead, yet has been found to achieve up to 70% compression [21].

4. BLOCKING SCHEMES

Among many variations, in our implementation, we use token blocking [5] – every pair of records becomes a candidate pair if they share at least one token. A separate block is associated with every token, containing records in which the token appears.

Phase 2 Blocking. The structure of our hash signatures enables augmenting the blocking process with a second phase executed on records of the same block to further reduce the set of candidate pairs. We find that the well-known Jaccard [5] metric is readily compatible. Jaccard is computed on a binary representation of a record's hash signature, such that non-zero weights are treated as a binary 1

and zero weights as a binary 0. Let D_A and D_B be the binary representations of A_x and B_y, respectively, the Jaccard metric is given by,

$$Jaccard(D_A, D_B) = \frac{|D_A \cap D_B|}{|D_A \cup D_B|} \quad (6)$$

The use of Jaccard for blocking appears to further increase performance, because it is computed very efficiently. The details are investigated in our experiments.

Information Leakage. The level of information leakage is evaluated for our protocol with respect to each blocking scheme introduced. It is important to note, however, there are subtle differences between the levels of sharing defined. To better illustrate this, consider a database of records mapped to hash signatures. It is easy to see that DB_{size} is divulged as it equals the number of hash signature entries. However, only **inf** Rec_{len} is divulged because a hash signature may incur collisions. Furthermore, **inf** $Vocab_{size}$ is revealed because the number of different entries in the set of hash signatures necessarily means different tokens. However, Tok_{frq} is not revealed because collisions do not preserve an accurate measure of the frequency of tokens.

On the other hand, if a subset of the hash signatures is available, only **inf** DB_{size} is divulged. However, Rec_{len} is not divulged, because it is an inaccurate measure (due to collisions) that is revealed on a subset of all records. Lastly, $Vocab_{size}$ is also not revealed for the same reason.

Assumptions. We partially base our analysis of security on several assumptions. Records are non-empty and do not solely consist of *stop list tokens* – very commonly occurring tokens in a database. We further assume that stop list tokens are not interesting: a measure of a category is considered accurate, even if it fails to represent stop list tokens.

In addition, if two hash signatures possess a similar set of weights stored, this does not necessarily imply that the hash signatures are the same. They are only considered the same if the weights reside in the same array entries. The reason for this will become more apparent in the following subsections.

4.1 Baseline Approach

Assume two parties Alice and Bob wish to compute the matching problem on their databases A and B, respectively. A is a set of records $A_1, ..., A_n$ and B is a set of records $B_1, ..., B_m$. Alice and Bob share a secret key k that is tacitly used in the computation of hash signatures. Carol is the mediating third party. In the "baseline" scheme, the matching problem is computed without the use of any blocking. The interest of this scheme is purely experimental. Alice and Bob transmit sequences of the hash signatures of databases A and B to Carol. Carol computes the matching problem on all pairs of A and B, and identifies matching records to Alice and Bob. This scheme proceeds as follows:

1. For each $A_x \in A$, Alice computes $HS(A_x)$. Likewise, for each $B_y \in B$, Bob computes $HS(B_y)$.

2. Let A_{hs} denote all $HS(A_x)$ that Alice transmits to Carol. Let B_{hs} denote all $HS(B_y)$ which Bob transmits to Carol.

3. For each $D_A \in A_{hs}$, D_A is paired with every $D_B \in B_{hs}$ for computing of the distance metric.

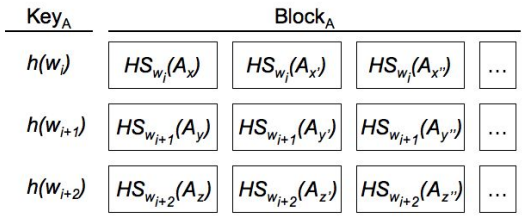


Figure 2: Phase 1 blocking in Simple blocking

This scheme does not employ any blocking, resulting in $O(nm)$ load. The additional categories of information revealed to Carol, I_0, in this scheme include: DB_{size}, **inf** $Vocab_{size}$, and **inf** Rec_{len} of databases A and B, because of the possibility of collisions in hash signatures. Tok_{frq} is not revealed.

4.2 Simple Blocking

In the Simple blocking scheme, token blocking is used to enhance the performance of the baseline scheme. The collaborating parties arrange records in blocks and transmit the blocks to Carol. The protocol steps are as follows:

1. For each $w_i \in A_x$, Alice computes $HS_{w_i}(A_x)$ and stores it in R_{w_i}, the block of w_i. Similarly, for each $w_j \in B_y$, Bob computes $HS_{w_j}(B_y)$ and stores it in S_{w_j}, Bob's block of w_j.

2. Let $Key_A \rightarrow Block_A$ denote all mappings $h(w_i) \rightarrow R_{w_i}$ that Alice transmits to Carol. Let $Key_B \rightarrow Block_B$ denote all mappings $h(w_j) \rightarrow S_{w_j}$, which Bob transmits to Carol.

3. For each $w \in Key_A$, if $w \in Key_B$, compute the Jaccard metric on every pair $D_A \in Block_A$ and $D_B \in Block_B$.

4. If $Jaccard(D_A, D_B) > threshold$, D_A and D_B are paired for computing of the distance metric.

In step 2 of this scheme, h is a keyed hashing function that provides confidentiality to the block identifier, i.e. the token; the key is concatenated analogous to hash signatures. Any secure hashing algorithm may be used. In our case, we implement SHA-1.

I_1, the additional categories of information divulged to Carol in Simple Blocking include: $Vocab_{size}$ because token blocking generates a separate block for each token in a record, and **inf** DB_{size} of databases A and B. Further, since Carol has no way of knowing whether hash signatures appearing in different blocks with permuted weights represent the same original record, only **inf** DB_{size} is determined by Carol. For the same reason and because of hash signature collisions, only **inf** Rec_{len} is divulged.

Finally, Tok_{frq} of databases A and B is divulged to Carol. Each block contains a list of hash signatures where the token appears, thereby an accurate measure of frequency of a token is revealed. This may prove to be a vulnerability to statistical attacks, such as Zipf [21] distribution. A technique to thwart such attacks is considered in Section 6.

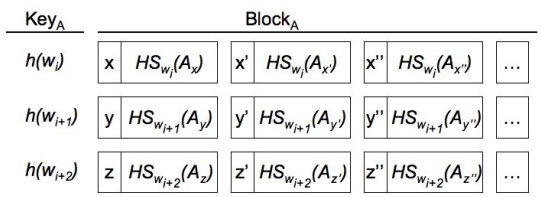


Figure 3: Phase 1 blocking in Record-aware blocking

Scheme	Information Categories			
	DB_{size}	$Vocab_{size}$	Rec_{len}	tok_{frq}
Baseline	Yes	**inf**	**inf**	No
Simple	**inf**	Yes	**inf**	Yes
Record-aware	Yes	Yes	Yes	Yes
Frugal Third Party	**inf**	**inf**	No	**inf**

Table 3: Summary of information leakage

4.3 Record-aware Blocking

In Simple blocking, the assignment of hash signatures to blocks equivocates a record across blocks where it is placed. The reason is that in step 1 of Simple blocking, the hash signature is computed with respect to the block (HS_{w_i}). Hence, the same record is indistinguishable to the third party when placed in multiple blocks. This means Simple blocking will compute duplicate distance metrics on the same pair of records proportional to the number of common blocks in which they appear. A relatively high ratio of common blocks per record pair can render the blocking scheme extremely inefficient. Record-aware resolves this by coupling an id with the hash signature of each record. The ids of every pairing made are maintained to ensure uniqueness of a pair. The blocking steps of Record-aware are as follows:

1. For each $w_i \in A_x$, Alice computes $HS_{w_i}(A_x)$ and stores the pair $[x, HS_{w_i}(A_x)]$ in R_{w_i}, the block of w_i. Similarly, for each $w_j \in B_y$, Bob computes $HS_{w_j}(B_y)$ and stores the pair $[y, HS_{w_j}(B_y)]$ in S_{w_j}, Bob's block of w_j.

2. Let $Key_A \rightarrow [ID_A, Block_A]$ denote all mappings $h(w_i) \rightarrow R_{w_i}$ that Alice transmits to Carol. Let $Key_B \rightarrow [ID_B, Block_B]$ denote all mappings $h(w_j) \rightarrow S_{w_j}$, which Bob transmits to Carol.

3. For each $w \in Key_A$, if $w \in Key_B$, Carol computes the Jaccard metric on every pair $D_A \in Block_A$ and $D_B \in Block_B$, such that $(ID_A, ID_B) \notin Paired$. $Paired$ is used to ensure that duplicate distance metrics on the same pair are not computed.

4. If $(ID_{D_A}, ID_{D_B}) \notin Paired$ and $Jaccard(D_A, D_B) > threshold$, D_A and D_B are paired for computing of the distance metric. Update $Paired = Paired \cup \{(ID_{D_A}, ID_{D_B})\}$

The increase in efficiency in this scheme is attained at the expense of additional data leakage in comparison to Simple Blocking. This scheme divulges all designated categories of information to Carol. Unlike Simple blocking, Carol determines DB_{size} of databases A and B, because an id is coupled with each hash signature that uniquely identifies the record which it represents. Rec_{len} are also divulged for the same reasons. As in Simple blocking, $Vocab_{size}$ and Tok_{frq} are revealed. This scheme provides the greatest exposure in terms of information leaked to Carol.

4.4 Frugal Third Party Blocking

In the previous two schemes, Alice and Bob transmit all of their blocks to Carol, even if only a fraction of the blocks are similar. Alternatively, Alice and Bob may find what blocks they have in common and only transmit those blocks. This arrangement has the advantage of minimizing communication size between participants. Further, the amount of data storage maintained by Carol for the matching problem is reduced, and Carol is relieved from the task of finding intersection between the blocks, resulting in less blocking and matching time. Moreover, from a monetary cost standpoint, any computation performed by Carol is characterized with a much greater price than computations performed by Alice or Bob. Carol is merely a mediating party which may charge a significant fee for facilitating matching between collaborating parties. It follows that it may be more cost-effective for Alice and Bob to perform as much computation as possible, even if such computation is orders of magnitudes greater than a logically equivalent computation performed by Carol.

Frugal Third Party blocking employs the secure intersection algorithm [23, 6] to compute the intersection set size using commutative one-way hash functions. In our implementations, we use RSA private keys with a common modulus. The scheme steps are as follows.

1. For each $w_i \in A_x$, Alice computes $HS_{w_i}(A_x)$ and stores the pair $[x, HS_{w_i}(A_x)]$ in R_{w_i}, the block of w_i. Similarly, for each $w_j \in B_y$, Bob computes $HS_{w_j}(B_y)$ and stores the pair $[y, HS_{w_j}(B_y)]$ in S_{w_j}, Bob's block of w_j.

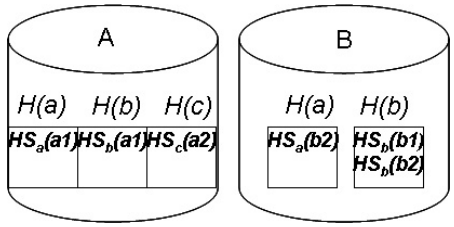
2. Let $Key_A \rightarrow [ID_A, Block_A]$ denote all mappings $h(w_i) \rightarrow R_{w_i}$. Let $Key_B \rightarrow [ID_B, Block_B]$ denote all mappings $h(w_j) \rightarrow S_{w_j}$. Using the secure intersection algorithm, find $Key_A \cap Key_B$. Alice transmits to Carol all $w \rightarrow [ID_A, Block_A]$ and Bob transmit all $w \rightarrow [ID_B, Block_B]$, such that $w \in Key_A \cap Key_B$.

3. For each $w \in Key_A \cap Key_B$, Carol computes the Jaccard metric on every pair $D_A \in Block_A$ and $D_B \in Block_B$, such that $(ID_A, ID_B) \notin Paired$. As in Record-aware, $Paired$ is used to ensure that duplicate distance metrics on the same pair are not computed

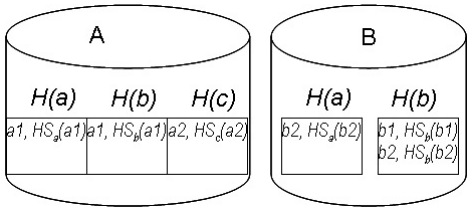
4. Finally, if $(ID_{D_A}, ID_{D_B}) \notin Paired$ and $Jaccard(D_A, D_B) > threshold$, D_A and D_B are paired for computing of the distance metric. Update $Paired = Paired \cup \{(ID_{D_A}, ID_{D_B})\}$

In this scheme, some additional categories of information are revealed only to Alice and Bob, while others only to Carol. Let I_{3AB} denote the additional categories revealed to Alice and Bob and I_{3C} denote categories revealed to Carol.

Alice and Bob predetermine the blocks which are shared by them. Hence, I_{3AB} contains $Vocab_{size}$, meaning Alice learns the vocabulary size of database B and Bob learns the vocabulary size of Alice. Since Carol does not receive all blocks, I_{3C} includes **inf** DB_{size} and **inf** $Vocab_{size}$ because an accurate measure is determined on a subset of the categories. Further, Carol only determines an accurate Tok_{frq}



(a) Blocks of Simple blocking



(b) Blocks of Record-aware and Frugal third party

Figure 4: Blocking Example

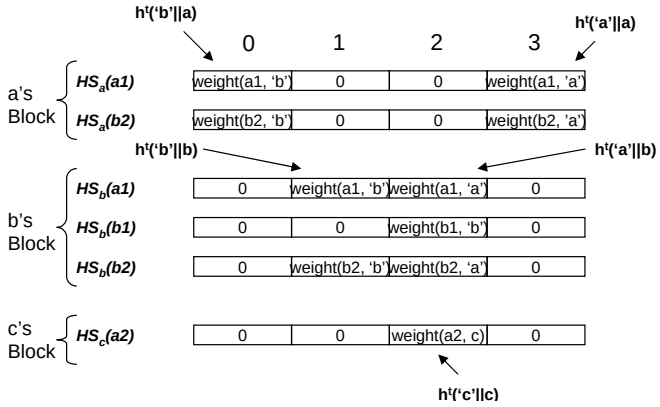


Figure 5: Hash signature contents

of a subset of all tokens because she only receives a subset of all blocks, hence **inf** Tok_{frq}. However, Carol does not determine Rec_{len} because an inaccurate measure is possible on merely a subset of records. The security of this scheme introduces a level of uncertainty to the categories divulged to Carol at the cost of increased computational cost and exposure of some categories to Alice and Bob. From an information leakage standpoint, Frugal Third Party is more useful when security with respect to the third party is of prime concern.

Example 3. Here, we present an example that illustrates the working of Simple, Record-aware, and Frugal Third Party blocking schemes. We consider databases A and B as defined in Table 1.

1. **Step 1.** Figure 4(a) illustrates how a record's hash signature is arranged in blocks that correspond to the tokens contained in the record. For example, record a1 contains tokens 'a' and 'b'; hence, it appears in the block corresponding to a as HS_a and the block corresponding to b as HS_b. Notice that a hash signature representation appears differently depending on the block where it is stored. This is accomplished by a concatenation of the block's token to the hash function used to generate the hash signature (note a key is also concatenated).

   The blocks of Record-aware and Frugal Third Party blocking schemes appear a little differently, as illustrated in Figure 4(b). As mentioned above, an id is coupled with each hash signature, so identical pairs are not matched for similarity multiple times. Assuming t is 2-bits, and $h^t('a'||a) = 3$, $h^t('b'||a) = 0$, $h^t('c'||c) = 2$, $h^t('a'||b) = 2$, $h^t('b'||b) = 1$, The hash signatures are computed as shown in Figure 5.

2. **Step 2.** In Simple and Record-aware blocking, Alice and Bob readily transmit their blocks of hash signatures computed in the last step to Carol. Alice transmits 3 blocks to Carol, while Bob transmits 2 blocks to Carol. However, in Frugal Third Party, Alice and Bob first determine the matching block names using a secure intersection algorithm. It is determined that only two blocks match, the block corresponding to a and b. Hence, Alice and Bob both transmit 2 blocks to Carol.

3. **Step 3.** Carol computes the distance metric on hash signatures which appear in common blocks. In Simple blocking, all records of the parties which appear in similar blocks are considered. This results in the following pairings: ($HS_a(a1)$, $HS_a(b2)$), ($HS_b(a1)$, $HS_b(b2)$), and ($HS_b(a1)$, $HS_b(b1)$). On the other hand, Record-aware and Frugal Third Party only pair a1 and b2 once, instead of twice. Hence, only two pairings are made: ($HS_a(a1)$, $HS_a(b2)$) and ($HS_b(a1)$, $HS_b(b1)$).

4. **Step 4.** The Jaccard metric is computed on a binary equivalent of each hash signature. Figure 6 illustrates binary equivalents of the hash signatures. If follows:

   - $Jaccard(HS_a(a1), HS_a(b2)) = 1$
   - $Jaccard(HS_b(a1), HS_b(b2)) = 1$
   - $Jaccard(HS_b(a1), HS_b(b1)) = .5$

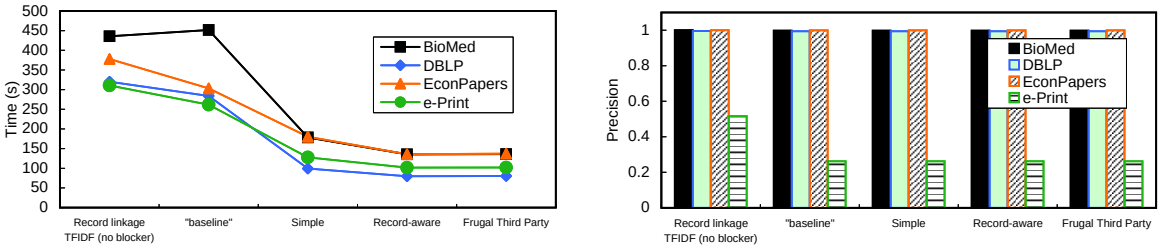
   In the absence of blocking, 4 distance metrics are calculated. If Jaccard $threshold > .5$, then 2 distance metric are calculated in Simple blocking, resulting in 50% reduction ratio. Only 1 metric is calculated in the Record-aware and Frugal Third Party schemes resulting in a reduction ratio of 25%. □

	0	1	2	3
$HS_a(a1)$	1	0	0	1
$HS_a(b2)$	1	0	0	1
$HS_b(a1)$	0	1	1	0
$HS_b(b1)$	0	0	1	0
$HS_b(b2)$	0	1	1	0
$HS_c(a2)$	0	0	1	0

Figure 6: Binary representations of hash signatures

Dataset	Domain	A size	B size	A ∩ B size	# of tokens per record (avg)
BioMed	Medicine	5000	5000	2000	25.44
DBLP	CompSci	5000	5000	2000	14.68
EconPapers	Economics	5000	5000	4000	21.23
e-Print	Physics	5000	5000	2000	13.83

Table 4: Experimental Datasets 1



(a) Total time (blocking + matching)

(b) Precision

Figure 7: Comparison of blocking + matching time of four implementations

.

5. EXPERIMENTAL VALIDATION

We have implemented our schemes based on the SecondString package [5], and have tested it using a range of datasets consisting of citation data. Each citation is parsed as a set of fields and only a random permuted subset of the fields are stored to achieve a level of heterogeneity. The choice of citation data was influenced by Ravikumar et al [6]. As an evaluation metric, *precision* is defined as the fraction of pairs matched by the distance metric that are correct, and *reduction ratio* as the ratio between candidate pair count from blocking and from a pair-wise comparison.

5.1 Blocking

Figure 7(a) proves our claims of increased performance due to blocking, by comparing the blocking schemes against the baseline and record linkage TFIDF that doesn't use blocking. Note, computation time due to the secure intersection algorithm in the Frugal Third Party scheme has been factored out. Table 4 lists the datasets used in this experiment and their characteristics. For all cases, the blocking techniques require a fraction of the time to solve the same problem, often at a negligible cost of precision. Figure 7(b) illustrates the precision observed for each dataset. The precision observed for the e-Print dataset deviates from other datasets. However, even in the non-private protocol, low precision is observed. Nonetheless, this is magnified in the private record linkage protocols.

Moreover, it is observed that representing records as hash signatures increases performance. We attribute this to the way hash signatures are represented, which is more compact than weight vectors in record linkage.

5.2 Blocking Schemes

To verify the properties of the proposed blocking scheme, several experiments were conducted using DBLP citations as in Table 5.

In Figure 8(a), the total time expended by the third party is measured as a function of dataset used. Simple blocking is the least efficient for the reason that every pair of records

Dataset	A size	B size	A ∩ B size
1	2500	2500	0
2	2500	2500	100
3	2500	2500	250
4	2500	2500	500
5	2500	2500	1000
6	2500	2500	2500

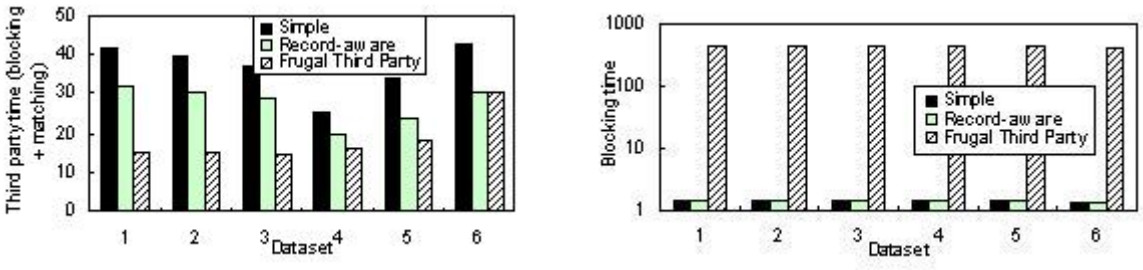
Table 5: Experimental Datasets 2

may be measured for similarity more than once. The total time expended for Simple and Record-aware blocking does not follow any specified pattern because of additional factors that come into play, such as the total number of blocks in each database, or the number of shared blocks. On the other hand, in Frugal Third Party, as the number of shared records increases, total third party time also increases because only common blocks are transmitted over to the third party. Clearly, the number of common blocks is likely to increase relative to the number of shared blocks. When A and B have the same records, no third party gains are observed by Frugal Third Party over Record-aware, resulting in transmission of almost all blocks, congruent to Simple and Record-aware.

Frugal Third Party reduction of third party time and restriction on privacy divulged to third party comes at an overwhelming cost of collaborating party blocking. Figure 8(b) illustrates the average blocking time of Frugal Third Party, Simple, and Record-aware schemes. It eludes to 4 orders of magnitude difference in blocking time.

The reduction ratio observed by two phase blocking is dependent on characteristics of the dataset, such as $Vocab_{size}$ and Tok_{frq}. Nonetheless, reduction ratio is directly responsible for the matching time needed to solve the problem. Figure 9 illustrates the linear relationship between reduction ratio and matching time.

5.3 Two-phase Blocking



(a) Total third party time

(b) Blocking time of collaborating parties

Figure 8: Comparison of three blocking alternatives

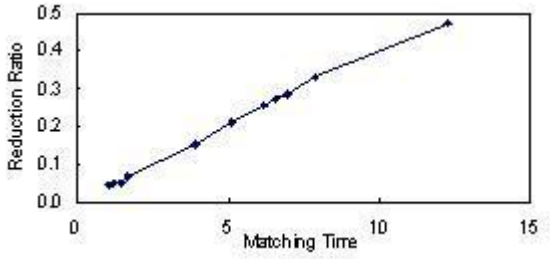


Figure 9: Relationship between reduction ratio and matching time

The second phase of blocking, Jaccard metric, depends on basic set operations to further reduce candidate pairs that undergo TFIDF scoring. The *threshold* used for Jaccard blocking is tunable for enhanced control over the level of blocking achieved. In Figure 7(a), the gains resultant from predominantly phase 1 blocking are observed. In practice, using phase 2 Jaccard, we are able to control the gains by adjusting the *threshold*. Figure 10 illustrates the performance increase when the Jaccard *threshold* is increased to .25. Experiments of the previous subsection all used *threshold* = .1.

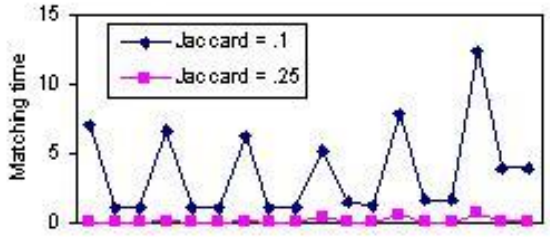


Figure 10: Relationship between reduction ratio and matching time for Dataset 1-6

6. OTHER ISSUES

The privacy of our protocol can be further increased by using techniques such as chaffing and winnowing [20]. Chaffing refers to the act of augmenting data with additional dummy data, indiscernible from other data, i.e. noise. Winnowing is the reverse process that eliminates dummy from real data. Chaffing is performed by the collaborating parties prior to transmitting data to the third party. After the results are computed and returned to the collaborating parties, all dummy data is winnowed appropriately.

Chaffing and winnowing techniques address the problem introduced by a Zipf [21] analysis of data blocks that we identified earlier. The problem is solved by fixing the content size of each block to make them appear indifferent, through the addition of noise. Nevertheless, this comes at the expense of increased cost and memory requirements. Our current implementations do not consider chaffing and winnowing.

7. CONCLUSION

In this paper, we described a secure protocol for record linkage and several schemes that achieve secure blocking. In the protocol, the collaborating parties use TFIDF to independently calculate and generate weight vectors, represented as hash signatures. This protocol requires a key to be negotiated by the collaborating parties, unknown to the third party.

The blocking schemes described are characterized by varying privacy as per minimum information sharing at the expense of performance. We define a loose characterization of information leakage to analyze the privacy of our protocols as a function of blocking scheme. Simple blocking arranges hash signatures in blocks, but a pair may be computed for similarity more than once if they are located in more than one common block. Record-aware solves this issue by coupling an identifier with every hash signature. Frugal Third Party provides the highest level of privacy with respect to information divulged to the third party, but requires the use of public-key dependent secure intersection algorithm, resulting in heavy computational costs on the collaborating parties. The performance of our blocking schemes was further enhanced by introducing a second phase of blocking: Jaccard.

Overall, parallel to record linkage, the use of blocking in private record linkage to enhance performance is validated using analytic and empirical validations. In some cases, blocking increases the security en lieu of performance of the protocol.

Future Work. It is interesting to investigate how other distance metrics and blocking schemes can be ported to the secure matching arena. In [6], the authors address other distance metrics and how they may be ported into secure record linkage. Clearly, similar techniques as described in

this paper may also apply to those metrics.

8. REFERENCES

[1] Rakesh Agrawal, Alexandre Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *Proceedings of ACM SIGMOD*, pages 86–97, 2003.

[2] Rohan Baxter, Peter Christen, and Tim Churches. A comparison of fast blocking methods for record linkage. In *Proceedings of 9th ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Consolidation*, 2003.

[3] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC 96*, pages 639–648, 1996.

[4] Tim Churches and Peter Christen. Some methods for blindfolded record linkage. *BMC Medical Informatics and Decision Making*, 4(9), 2004.

[5] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for matching names and records. In *KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[6] William Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A secure protocol for computing string distance metrics. In *Proceedings of ICDM Workshop on Privacy and Security Aspects of Data Mining*, 2004.

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

[8] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of SIGMOD*, 2003.

[9] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. In *Proceedings of 8th ACM SIGKDD*, pages 217–228, 2002.

[10] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[11] Luis Gravano, Panagiotis G. Ipeirotis, Jagadish Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava. Approximate string joins in a database (almost) for free. In *Proceedings of 27th VLDB*, pages 491–500, 2001.

[12] Luis Gravano, Panagiotis G. Ipeirotis, Koudas Koudas, and Divesh Srivastava. Text joins in an rdbms for web data integration. In *Proceedings of 12th WWW*, January 01 2003.

[13] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In *Proceedings of ACM SIGMOD*, pages 127–138, 1995.

[14] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random data perturbation techniques and privacy preserving data mining. In *Proceedings of ICDM*, pages 160–164, 2003.

[15] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay — a secure two-party computation system. In *Proceedings of 11th USENIX Security Symposium*, August 2004.

[16] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954959, 1959.

[17] Huseyin Polat and Wenliang Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of ICDM*, 2003.

[18] Catherine Quantin, H. Bouzelat, F. Allaert, A. Benhamiche, J. Faivre, and L. Dusserre. How to ensure data security of an epidemiological follow-up: quality assessment of an anonymous record linkage procedure. *International Journal of Medical Informatics*, 49(1):117–122, 1998.

[19] Catherine Quantin, H. Bouzelat, and L. Dusserre. A computerized record hash coding and linkage procedure to warrant epidemiological follow-up data security. *Studies in Health Technology and Informatics*, 43:339–342, 1997.

[20] Ronald Rivest. Chaffing and winnowing: Confidentiality without encryption. *MIT, Internal Paper*, 1998.

[21] Gerard Salton, editor. *Automatic Text Processing.* Addison Welsley, 1989.

[22] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[23] Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to associate rule mining. *Journal of Computer Security*, 2004. To Appear.

[24] W. E. Winkler. Matching and record linkage. *Business Survey Methods*, pages 355–384, 1995.

[25] L. Xiong, S. Chitti, and L. Liu. Topk queries across multiple private databases. In *25th ICDCS. To appear*, 2005.

[26] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Symposium on FOCS*, pages 160–164, 1982.