Frequent *grams* based Embedding for Privacy Preserving Record Linkage

Luca Bonomi
Emory University
Atlanta, USA
lbonomi@mathcs.emory.edu

Li Xiong
Emory University
Atlanta, USA
lxiong@mathcs.emory.edu

Rui Chen
Concordia University
Montreal, Canada
ru_che@encs.concordia.ca

Benjamin C. M. Fung
Concordia University
Montreal, Canada
fung@ciise.concordia.ca

ABSTRACT

In this paper, we study the problem of privacy preserving record linkage which aims to perform record linkage without revealing anything about the non-linked records. We propose a new secure embedding strategy based on frequent variable length grams which allows record linkage on the embedded space. The frequent grams used for constructing the embedding base are mined from the original database under the framework of differential privacy. Compared with the state-of-the-art secure matching schema [15], our approach provides formal, provable privacy guarantees and achieves better scalability while providing comparable utility.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*

General Terms

Algorithms, Security, Performance

Keywords

Privacy, Security, Record Linkage, Differential Privacy

1. INTRODUCTION

Record linkage [18, 7] plays a central role in many data integration and data mining tasks that involve data from multiple sources. It is the process of identifying records that refer to the same real world entity across different sources. It is extensively used in many applications, for example, in linking medical data of the same patient across different hospitals in the country or in collecting the credit history of users from several sources. However, many of these data may contain sensitive personal information that could disclose individual privacy. For this reason, the problem of privacy preserving record linkage has drawn considerable attention over recent years. The objective is to allow two parties to identify records that are close to each other according to some distance function, such that no additional information about the data records other than the result is disclosed to any party.

In the existing literature, several techniques have been proposed, and they can be mainly categorized into a few categories: Secure Multiparty Computation (SMC) [13, 20], secure transformation [1, 4, 15, 16], and hybrid methods [8, 10, 11, 19]. SMC techniques use cryptographic mechanisms and allow two parties to perform record linkage as a secure function such that no party knows anything except its own input and the results. However, they are computationally prohibitive in practice. Secure transformation methods use data transformation techniques such as one-way hashing or embedding to map the original data into new data values that cannot be reversed and then perform record linkage on the transformed data typically by a third party. While these methods are more efficient, the challenge is to have a secure transformation while preserving the accuracy of linkage on the transformed space as high levels of protection typically implies a great loss of accuracy in the final results. Finally, hybrid techniques attempt to combine anonymization or transformation techniques with SMC protocols. They typically use a privacy preserving *blocking* step to restrict the comparisons to smaller groups of records which are then compared by SMC protocols. While they provide a trade-off between efficiency and accuracy, the SMC step is still required and are typically not implemented or evaluated due to the high computation cost.

In this paper, we present a new secure data transformation method based on frequent grams embedding and we provide a comparison with the approach proposed by Scannepieco et. al. [15] for linking string records. The latter uses SparseMap [9] to embed strings into a vector space, where the common base among the parties is formed by random strings, so that no information is disclosed. However, as the protocol is designed, the shared base is optimized according to the data at one party. Therefore, if a malicious party is involved, some sensitive information could be disclosed. In contrast, our protocol uses frequent grams as a base for secure embedding which gives a better representation of the records than a random base and the frequent grams are mined from the original database with a formal guarantee of *differential privacy* [5]. As a proof-of-concept, we focus on string records in this paper, and perform approximate matching of the records based on a similarity criterion.

Our contributions:

- We propose a novel embedding strategy based on frequent variable length grams to map string records into vectors in the real space. We show that the use of frequent variable length grams substantially increases the utility of the results with respect to random bases.

- We adapt and extend the privacy preserving mining algorithm in [3] to mine frequent variable length grams which can be used as the embedding base. The proposed privacy preserving record linkage protocol hence satisfies the differential privacy framework which provides formal guarantees of individual privacy.
- Finally, we present a set of empirical experiments using real world datasets showing the benefit of our approach.

The rest of the paper is organized as follows. In Section 2, we introduce some basic definitions and the privacy model adopted in our solution. Section 3 provides a description of the major components in our proposed solution. The experiment results are reported in Section 4. Finally, we conclude the paper in Section 5

2. PRELIMINARIES

In this section, we introduce some notations and definitions related to our approach.

Let Σ be a finite alphabet, we denote by $x = x_0 x_1 \cdots x_{n-1}$ a string of length n where each symbol x_i is defined in Σ. Moreover, we denote by $|x|$ the length of the string x. As a similarity measure between strings we consider the Edit distance [12] (d_{Edit}), which measures the number of edit operations needed to transform a string into the other one. The problem of record linkage that we consider in this paper is defined as follows.

PROBLEM 1 (RECORD LINKAGE). *Given two sets D_A and D_B of string records, find $\mathcal{M} \subset D_A \times D_B$, such that $\mathcal{M} = \{(x, y) \mid d_{Edit}(x, y) \leq ed\}$ (matching records) and no information about the individual records $(x, y) \notin \mathcal{M}$ (non-matching records) is disclosed.*

2.1 Differential Privacy

Differential privacy [5] is a recent notion of privacy that aims to protect the disclosure of information when statistical data are released. The differential privacy mechanism guarantees that the computational output is insensitive to change in any particular individual record of the input data.

DEFINITION 1 (DIFFERENTIAL PRIVACY [6]). *A non interactive privacy mechanism M has ϵ-differential privacy if for any two input sets (databases) D_A and D_B with symmetric difference one (neighbor databases), and for any set of outcomes $S \subseteq Range(M)$,*

$$Pr[M(D_A) \in S] \leq exp(\epsilon) \times Pr[M(D_B) \in S] \qquad (1)$$

where ϵ is the privacy parameter (also referred to as privacy budget). Intuitively, lower value of ϵ implies stronger privacy guarantees, and vice versa.

Two composition properties are extensively used when multiple differential privacy computations are combined. These two properties are known as *sequential* and *parallel* compositions [14]. The former states that any sequence of computations that each provides differential privacy in isolation also provides differential privacy in sequence. The latter instead holds when the computations involved are performed on disjoint data. In this case, the privacy cost does not accumulate but depends only on the worst guarantee.

A common mechanism to achieve differential privacy is the Laplace mechanism [6] which we will use in this paper. Let f be a statistical function, and ϵ be the privacy parameter, the mechanism adds calibrated noise to the result $f(D)$ in order to guarantee ϵ-differential privacy. The noise is generated from a Laplace distribution with probability density function $pdf(x|\lambda) = \frac{1}{2\lambda} e^{-|x|/\lambda}$, where the parameter λ is determined by ϵ and $GS(f)$, the *global sensitivity* [6]

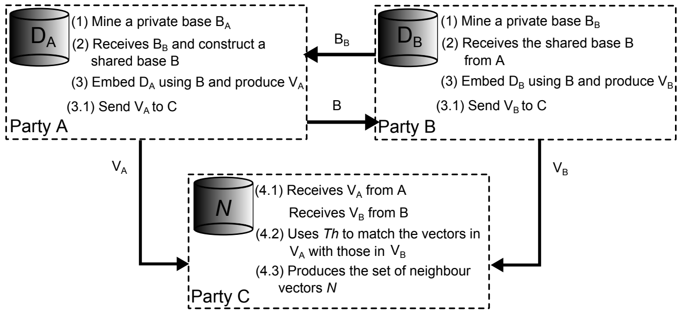

(1) Mine a private base B_A
(2) Receives B_B and construct a shared base B
(3) Embed D_A using B and produce V_A
(3.1) Send V_A to C
Party A
D_A
B_B
B
(1) Mine a private base B_B
(2) Receives the shared base B from A
(3) Embed D_B using B and produce V_B
(3.1) Send V_B to C
Party B
D_B
V_A
V_B
N
(4.1) Receives V_A from A
Receives V_B from B
(4.2) Uses Th to match the vectors in V_A with those in V_B
(4.3) Produces the set of neighbour vectors N
Party C

Figure 1: Overview of the Secure Protocol.

of the function to the inclusion and exclusion of any record in the dataset. We restrict our attention to counting queries, which can be proven to have $GS(count) = 1$.

3. OUR SOLUTION

3.1 Overview

We propose an embedding technique based on *grams*, which allows approximate matching of the records (i.e. within a fixed number of edit operations). The proposed technique maps the original data into a vector space by projecting each string in the databases on a base formed by a set of frequent *grams*, where a gram of length q is a substring $x_0 x_1 \cdots x_{q-1}$ of the original strings.

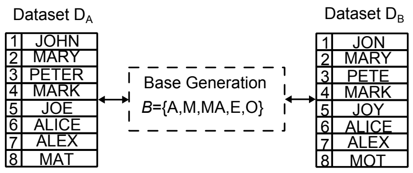
Each party starts to build a base for the embedding by mining grams from its own database, and this phase is denoted as **mining phase**. This process is performed with a guarantee of differential privacy, so that the parties involved in the protocol can share their bases and determine a common base for the embedding without disclosing any sensitive information of individual records. When a final base is determined, each party embeds its data using the common base, and the matching is performed in the embedded space. We denote this step as **embedding phase**. Our overall protocol is illustrated in Figure 1. Our strategy requires the presence of a third trusted party denoted by C, whose task consists in matching the records in the embedded space. A summary of the steps is listed as follows.

1. **Mining Phase:** Parties A and B apply a differentially private algorithm to mine their respective databases D_A and D_B, and compute *private* bases $\mathcal{B}_A$ and $\mathcal{B}_B$.
2. **Base Generation:** One of the two parties is in charge of merging the two bases and producing a shared base $\mathcal{B}$ of frequent variable length grams.
3. **Embedding Phase:** Each party A and B, by using the shared base, embeds its own data and generates a set of vectors V_A and V_B respectively, representing the strings in the original datasets. These sets are sent to the third party C.
4. **Matching Phase:** The third party C, for each vector $\bar{s} \in V_A$ returns a set of neighbor vectors $\mathcal{N}$ from V_B that are within Euclidean distance of Th (*global threshold*). This set identifies the matching set $\mathcal{M}$.

Figure 2(a) illustrates the mining and base generation phase. The party A and B mine their respective datasets and produce a shared private base formed by the following grams $\{\texttt{A}, \texttt{M}, \texttt{MA}, \texttt{E}, \texttt{O}\}$. This base is used to produce the set of embedded vectors in Figure 2(b).

3.2 Mining phase

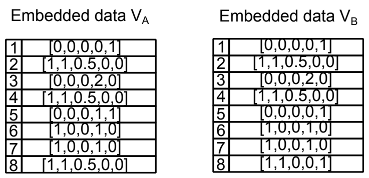
Contrary to the SparseMap approach in [15], we construct a base *mined* from the original data mainly for two reasons. First, we take



Dataset D_A	
1	JOHN
2	MARY
3	PETER
4	MARK
5	JOE
6	ALICE
7	ALEX
8	MAT

Dataset D_B	
1	JON
2	MARY
3	PETE
4	MARK
5	JOY
6	ALICE
7	ALEX
8	MOT

(a) **Mining and Base Generation**



Embedded data V_A	
1	[0,0,0,0,1]
2	[1,1,0.5,0,0]
3	[0,0,0,2,0]
4	[1,1,0.5,0,0]
5	[0,0,0,1,1]
6	[1,0,0,1,0]
7	[1,0,0,1,0]
8	[1,1,0.5,0,0]

Embedded data V_B	
1	[0,0,0,0,1]
2	[1,1,0.5,0,0]
3	[0,0,0,2,0]
4	[1,1,0.5,0,0]
5	[0,0,0,0,1]
6	[1,0,0,1,0]
7	[1,0,0,1,0]
8	[1,1,0,0,1]

(b) **Embedded Data**

Figure 2: Base Generation and Embedding of the Data.

advantage of the fact that the strings being matched in record linkage scenarios typically have similar properties (e.g. same alphabet, similar length, etc.). Hence, by using a base mined from the original dataset, we can capture this information. Second, a randomly generated base can not represent every dataset well since it is defined in a generic way and not data dependent.

We form a base by mining the *frequent* grams in the database. Formally, given a positive integer k, a minimum length q_{min} and a maximum length q_{max}, our goal is to mine the top-k frequent q-grams where $q \in [q_{min}, q_{max}]$, and to use this set as a base for the embedding. Intuitively, we can obtain a base set that is a good representative for all the strings in the databases since frequent grams are more likely to be shared among the strings. In addition, by restricting the attention to the top-k frequent grams, we can control the dimensionality of the data in the new space. In order to protect the privacy of individual records, the grams are mined from the original dataset to guarantee ϵ-differential privacy. In our approach, we consider an extension of the *prefix tree* mining algorithm [3] originally introduced to mine frequent trajectory data. In our case, the two databases may be correlated as they may contain records belonging to the same entity, the total privacy parameter is split among the two parties holding the dataset, so that the overall privacy level is ϵ.

3.2.1 Prefix-tree Miner

The mining algorithm proposed in [3] mines the frequent trajectory data by using a prefix tree. In the same way, we partition the space of all the possible grams using a top-down approach, where each partition is identified by a node in a prefix tree $\mathcal{T}$. Each node has the following information: a prefix ω, an accumulated privacy budget, and the subset of all the strings in the original database having ω as a prefix, called the partition represented by the node.

The construction of the prefix tree can be summarized as follows. Starting from the root of the tree, the database is partitioned by extending the prefix of the current node using Algorithm 1. For every symbol a in the alphabet Σ, a new node is attached to the tree only if the string ωa is a frequent prefix, where ω is the prefix represented by the parent of the current node. To determinate if a prefix is frequent, a counting query is issued on the partition of the dataset represented by the current node and the real count is perturbed by Laplace noise to guarantee differential privacy. In this process, only partitions with frequent prefixes ($count > \theta$) are further refined. The allocation of the budget at each level in the tree is performed at line 6 in Algorithm 1. In our approach, we propose several strategies to allocate the private budget: *linear allocation*, *exponential allocation*, *adaptive*, and *hybrid*. Details

Algorithm 1 Private Prefix-Tree Partitioner

```
1: procedure PPT PART($node$, $\mathcal{T}$)
   Input: node $node$; private prefix-tree $\mathcal{T}$
   Output: $\mathcal{T}$ private Prefix-Tree

2:     while (($node.h \leq h_{MAX}$) && ($node.budget < \epsilon$)) do
3:         for (every symbol $a$ in the alphabet $\Sigma$) do
4:             $\omega \leftarrow$ (path from $r$ to node) + $a$
5:             $P \leftarrow \{x \in node.set$ s. t. $\omega$ is a prefix of $x\}$
6:             $\tilde{\epsilon} \leftarrow$ ALLOCATE BUDGET($node$)
7:             $count \leftarrow |P| + Lap(1/\tilde{\epsilon})$
8:             if ($count > \theta$) then                          ▷ Non empty node
9:                 Add a new node $cur$ as child of $node$, such that:
10:                    $cur.set \leftarrow P, cur.pdfilon \leftarrow \tilde{\epsilon}$
11:                    $cur.label \leftarrow \omega, cur.h \leftarrow node.h + 1$
12:                    $cur.budget \leftarrow cur.budget + cur.pdfilon$
13:                    PPT PART($cur$,$\mathcal{T}$)                ▷ Recursive call
14:            end if
15:        end for
16:    end while
17:    return $\mathcal{T}$
18: end procedure
```

about these strategies are presented later in this section. After we partition the data, we traverse the prefix tree and apply the consistency constraints for each root-to-leaf path as in [3]. Once the consistency constraints are enforced, we identify a list of frequent grams by traversing the tree. As a final result, we return the top-k grams sorted by their noisy frequencies. An example of the prefix tree is illustrated in Figure 3.

Budget Allocation Strategies: We investigate and propose more allocation strategies than the linear allocation introduced in [3].

- **Linear:** Each node at each level in the tree is allocated the same amount of budget.
- **Exponential:** At level i in the tree, a node is allocated a budget double the amount of its parent.
- **Adaptive:** This strategy is an adaptation of the previous exponential allocation strategy, where the entire remaining budget on the path is spent on the next counting query if the current node represents a non frequent prefix.
- **Hybrid:** This strategy is a combination of the previous strategies, where the total budget is distributed in the tree according to q_{max}. In particular, we reserve half of the total budget to the nodes on the first q_{max} levels of the tree, where the budget is allocated to each node in a linear fashion. For the remaining nodes, the adaptive strategy is used.

Privacy Analysis: All the partitions produced by Algorithm 1 on the same level of the tree are disjoint since they correspond to strings with different prefixes. Therefore, by the *parallel composition* property [14], the overall privacy level is determined by the maximum value of the budget used over all the root-to-leaf path of the tree. For any path, the overall privacy level is given by the *sequential composition* property [14] and can be computed as the sum of the privacy budget used for each counting query for each node on the path.

THEOREM 1 (PREFIXTREE ϵ- PRIVACY). *The Prefix-tree Miner guarantees ϵ-differential privacy.*

PROOF. *The proof follows directly from the differential privacy result proposed in [3] since all the allocation strategies uses at most ϵ budget on the root-to-leaf paths in the tree.* □

Complexity Analysis: Algorithm 1 has running time proportional to the number of nodes in the prefix tree $\mathcal{T}$. By using a similar analysis as in [3], it can be shown that our mining approach requires $O(N|\Sigma|^{h_{MAX}+1})$ operations, where N is the size of the dataset, Σ is the alphabet, and h_{MAX} is the maximum depth in the tree.

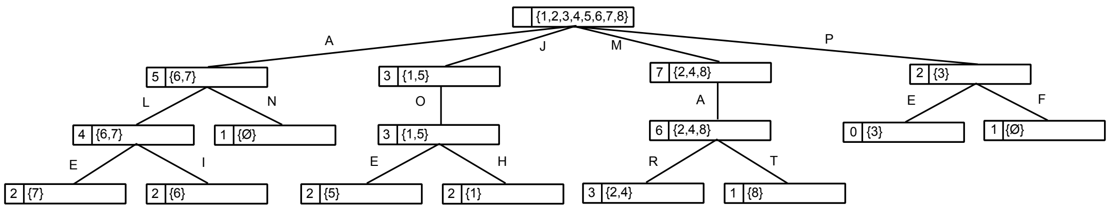


Figure 3: Prefix Tree Example from D_A in Figure 2(a). Each nodes has a noisy count, and the set of identifiers for the strings in the partition. Each branch of the tree is labeled with the symbol used to extend the prefix.

3.3 Embedding phase

In this section, we describe the embedding phase to map strings into vectors. Let $\mathcal{B} = \{g_1, g_2, \ldots, g_k\}$ be a base of k grams, each string s in the database is mapped into a vector $\bar{s}$ in $\mathbb{R}^k$, where each component $\bar{s}_i$ represents the number of occurrences of the gram g_i in s, normalized by the length of g_i. Let $Occ_s(g)$ denote the set of positions in s where the gram g occurs, then each coordinate is defined as $\bar{s}_i = |Occ_s(g_i)|/|g_i|$, for $i = 1, \ldots, k$. In this new space, the distance between vectors is computed using the Euclidean distance: $d'(\bar{x}, \bar{y}) = \|\bar{x} - \bar{y}\|_2$. An example of embedded records is illustrated in Figure 2(b).

Threshold Computation: In the original space, we are interested in matching strings within ed edit operations; however in the new space, this task is casted into the problem of finding all the vectors whose Euclidean distance is within a threshold value Th. This threshold value plays a central role on the overall performance, since it will determine the candidate records that may represent matching strings. Therefore, it is crucial to compute a threshold value as tight as possible to the real value. This problem is generally very hard, since proving formal guarantees requires analysis on the distance distortion and properties of the embedding strategy used. We define as a *global threshold* value, the Euclidean distance that can be used in matching all the records in the new space. This can be done by estimating how the original distance is distorted after the embedding map is applied. In this direction, we propose an initial upper bound for our embedding.

PROPOSITION 1 (UPPER BOUND). *Given x and y, two strings in the original space with Edit distance $d_{Edit}(x, y) \leq ed$, then $d'(\bar{x}, \bar{y}) \leq \Delta_q \cdot ed$, where $\Delta_q = q_{max} - q_{min} + 1$*

PROOF. *The number of grams of length q that can contribute to the distance is at most $q \cdot ed$ (since on a position i at most q grams of length q are overlapping). Since the base $\mathcal{B}$ used in the embedding is a subset of all variable length q-grams, it follows that: $d'(\bar{x}, \bar{y}) = \|\bar{x} - \bar{y}\|_2 \leq \Delta_q \cdot ed$.* □

In addition to this approach, we also studied the concept of *personalized threshold* which dynamically computes a threshold value for each individual record required to be matched. Due to space restrictions, we focus on global threshold in this paper and refer readers to [2].

4. EXPERIMENTAL RESULTS

In this section, we present a set of experimental results evaluating the impact of the private parameter ϵ and the dimensionality k on the overall utility of our protocol. We compare our approach with the method in [15] to show the benefit of our approach in scalability and stronger privacy model while achieving comparable data utility.

We use two real datasets, NAMES [1] and CITIES. The first contains a list of the most frequent surnames from the Census 2000.

[1]NAMES is publicly available at United States Census (http://www.census.gov/genealogy/www/data/2000surnames/)

Table 1: Experimental Datasets Statistics

Dataset	N	l_{max}	l_{min}	l_{avg}
NAMES	150000	15	4	7
CITIES	5000	23	3	8

Table 2: Experiment Parameters

Symbol	Description	Default values
ϵ	Private parameter	0.1
k	Size of the base	75
q_{min}	Min length of the grams	1
q_{max}	Max length of the grams	3
ed	Edit operations	$\{0, 1, 2\}$
h_{MAX}	Max depth of prefix tree	l_{avg}
θ	Threshold for noisy count	as in [3]
Th	Global threshold	$\{0, 1.3\}$

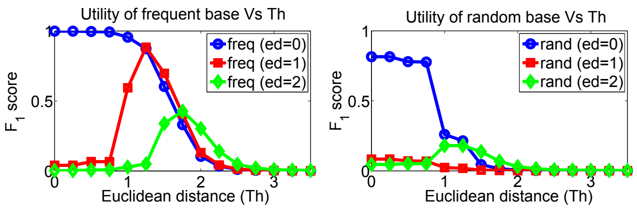
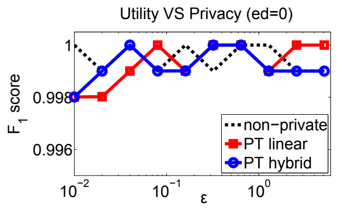
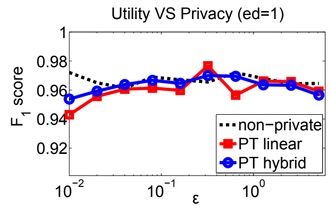


Figure 4: Utility and threshold: (Left) frequent grams, (Right) random grams

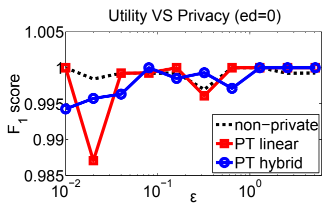
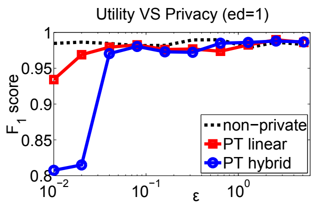
The second is a list of the top 5000 most populated cities in U.S. in 2008. Some of the statistics of the datasets are summarized in Table 1, where l_{max}, l_{min} and l_{avg} are the maximal, minimal and average lengths of the strings, respectively. The experiment and algorithm parameters, if not specified in the descriptions, assume the default values as reported in Table 2.

Frequent grams vs random grams: We first verify the advantage of using frequent grams over random grams in the embedding base. Figure 4 reports the utility in terms of F_1 score [17] for different values of global threshold (Th) in the embedded space, with a direct comparison between random and frequent grams. We tested the embedding strategy on the NAMES dataset, by allowing an approximate matching with the number of edit operations up to 2, with $k = 100$. From the graph, it is evident that frequent grams lead to a considerable improvement in the utility compared to random grams (an improvement from 20% to 60%). This result is justified by the fact that a base of frequent grams is more likely to share a higher number of grams with the strings. In addition, we can also observe that when approximate matching is allowed, the utility decreases as the number of edit operations increases.

Impact of the privacy parameter: The relationship between the privacy parameter ϵ and the utility of our protocol is reported in Figure 5. In these graphs, we also compare the results provided by private miners with an exact non private miner. We restrict our attention to the linear and hybrid budget allocation strategies since their performances are slightly better than those provided by the

(a) **Utility results with `NAMES` dataset**

(b) **Utility results with `CITIES` dataset**

Figure 5: Impact of the privacy parameter: (Left) Exact match, (Right) Approximate match.

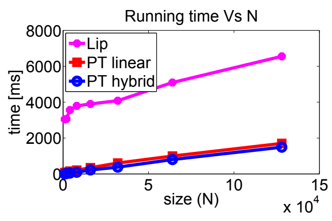


Figure 6: Running Time Vs N: comparison between the frequent variable length techniques and Lipschitz embedding.

other two strategies. As we can see, the utility of the protocols using private miners approaches the results obtained from the non private algorithm as ϵ increases. Moreover, this figure points out the hardness of solving record linkage when approximate matching is allowed. Indeed, the utility for approximate matching is moderately smaller than that of exact matching.

Protocol Performances: The scalability results of our strategy are reported in Figure 6. The running time is measured in milliseconds [ms], and it consists of the time needed to mine the base for each party, to combine the bases to form the shared base, and to embed the data. As we can see from Figure 6, the running time for our protocol is linear with the size of the dataset considered. Figure 6 shows also the running time for the Lipschitz approach proposed in [15]. For this approach, we measure the time required to generate the base using the heuristic and produce the embedding map. As we can see, this approach also scales linearly with the size of the dataset, but its running time is considerably higher. In Figure 7 we tested the approaches with different base sizes on the `CITIES` dataset. As we can see, Lipschitz provides a better utility for smaller bases, while our strategies achieve similar results when $k \geq 20$. However, the dependency of the running time with respect to the dimensionality is exponential for the Lipschitz strategy, while in our approach is considerably lower.

5. CONCLUSIONS

In this paper, we presented a novel frequent grams based embedding strategy to perform privacy preserving record linkage for string records. Compared with the state-of-the-art secure matching approach [15], our approach provides formal, provable privacy guarantees of differential privacy and achieves better scalability while providing comparable utility. As future work, we plan to enhance the allocation strategies for the prefix tree miner and the threshold schemes for matching in embedded space.

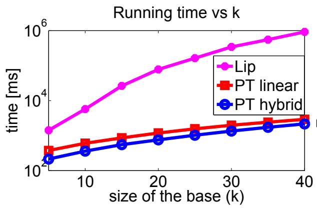
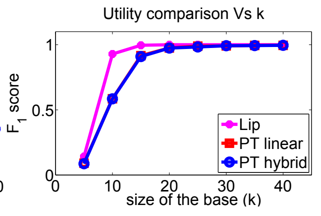



Figure 7: Performance of Lipschitz and frequent grams embedding Vs k. (Left) Running time, (Right) Utility

6. ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1117763. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

7. REFERENCES

[1] A. Al-Lawati, D. Lee, and P. McDaniel. Blocking-aware private record linkage. In *Proceedings of the 2nd international workshop on Information quality in information systems*, IQIS '05, pages 59–68, New York, NY, USA, 2005. ACM.

[2] L. Bonomi, L. Xiong, R. Chen, and B. C. M. Fung. Privacy preserving record linkage via grams projections. *eprint on arxiv.org*, 2012.

[3] R. Chen, B. C. M. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: A case study on the montreal transportation system. In *Proc. of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Beijing, China, August 2012. ACM Press.

[4] T. Churches and P. Christen. Some methods for blindfolded record linkage. *BMC Medical Informatics and Decision Making*, 4(1):9, 2004.

[5] C. Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006*, pages 265–284, 2006.

[7] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[8] L. O. Evangelista, E. Cortez, A. S. da Silva, and W. M. Jr. Adaptive and flexible blocking for record linkage tasks. *JIDM*, 1(2):167–182, 2010.

[9] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):530 – 549, may 2003.

[10] A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco. A hybrid approach to private record linkage. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 496–505, Washington, DC, USA, 2008. IEEE Computer Society.

[11] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private record matching using differential privacy. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 123–134, New York, NY, USA, 2010. ACM.

[12] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[13] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197, 2008.

[14] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM.

[15] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid. Privacy preserving schema and data matching. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 653–664, New York, NY, USA, 2007. ACM.

[16] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using bloom filters. *BMC Medical Informatics and Decision Making*, 9(1):41, 2009.

[17] C. Van Rijsbergen. *Information retrieval*. Butterworths, 1979.

[18] W. Winkler. Overview of record linkage and current research directions. Technical Report Statistics #2006-2, Statistical Research Division, U.S. Bureau of the Census, 2006.

[19] M. Yakout, M. J. Atallah, and A. Elmagarmid. Efficient private record linkage. *Data Engineering, International Conference on*, 0:1283–1286, 2009.

[20] A. C.-C. Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162 –167, oct. 1986.