T-79.514 Special Course on Cryptology / Privacy-Preserving Data Mining:

Secure Approximate Matching

Matti Järvisalo

1 Introduction

The goal of *pattern matching* is to decide whether a given object is among a group of objects (or part of a larger object). As an example, one might wish to find out whether a given fingerprint is in a database that contains fingerprints of thiefs. In *approximate* pattern matching one is interest in finding a member of a set of object that is approximately the same as a given object. It is often the case that exact pattern matching cannot be used, and in these cases approximate matching is the only way to go. As an example, although fingerprints are individual, in the process of digitizing a fingerprint we are forced to make only an approximate copy of the fingerprint.

The term *approximately* may have many interpretations depending on the application area, as the metrics used for measuring the distance between two objects vary depending on the application area; e.g., when matching images it's often reasonable to use the *Euclidean metric*, while in approximate string matching we might use the *edit distance* between two string as the metric. A hot topic in string matching is e.g. DNA sequence alignment.

There exist well developed solutions to both exact and approximate pattern matching for many domains. In many cases, traditional solutions are problematic for a completely different perspective, namely considering privacy concerns. Imagine the following scenario: a person, say Alice, wishes to compare her DNA sequence to well-known DNA sequences that cause genetic diseases. Alice, as a private individual, can only access a (public or non-public) DNA sequence database which is held by another individual, say Bob. Now, Alice certainly is not prepared to reveal the pattern matching results to Bob, as this might cause Alice troubles e.g. in finding a job given that she indeed has a genetic disease, nor is she ready to reveal her DNA sequence to Bob. In some cases Bob will also be interested in *not* learning Alice's query; although Bob might be trusted upon by Alice through a contract, Bob might get sued by Alice if private information q about Alice, trusted to Bob, becomes public.

To be a bit more formal, we are interested in finding good solutions that solve the following problem called *secure database access* (SDA):

> Alice has a string q, and Bob has a database of strings $T = \{t_1, \ldots, t_N\}$, all of which belong to some arbitrary set of strings. Alice wants to *match* q to the entries it T. Give a protocol that accomplishes this without revealing to Bob neither (i) q nor (ii) the match to the query.

Notice that in the definition of SDA above, we intentionally do not take stand on what is considered a *match*. This can thus be an exact or an approximate match, and different metrics can be considered. In the approximate case, the results that Alice gets can be either (i) the entry in T that is closest to q or (ii) the distance between q to the closest entry. In this survey, we consider solutions to SDA in the approximate case. We base our discussion mainly on a paper by W. Du and M.J. Atallah [1]. For the exact case, we refer the reader to e.g. [4]. It should be noted that although the SDA problem is a special case of secure multiparty computation [3], it is of interest to develop SDA-specific solutions, as they can in many cases be more *efficient*, enabling more practical methods for solving such problems.

2 Metrics of Interest

Let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ be two strings of length n. In this survey the set of metrics appearing in the discussion are the following.

- the first norm, i.e., $\sum_{i=1}^{n} |a_i - b_i|$;
- the second norm or the Euclidean metric, i.e., $\sum_{i=1}^{n} (a_i - b_i)^2$;
- a general metric, i.e. $\sum_{i=1}^{n} f(a_i, b_i)$, where f is a function (and thus e.g. the first and second norms are special cases of f); and
- edit distance [2], which is a measure on the cost of transforming one string into another. Usually the allowable transformation steps considered are *substitution* of a character with another character, *deletion* of a character, and *insertion* of a character.

The considered alphabet must vary with the metric, e.g., for edit distance we may consider an arbitrary alphabet, while with "numerical" metrics such as the second norm we expect that strings are encoded with numbers, i.e., the alphabet consists of natural numbers.

In [1] the authors also list as an additional metric the number of indices in which two strings differ. Oddly, though, they do not discuss solutions for such a metric apart from considering this metric a special case of the general metric.

3 Considered Models

We consider the SDA problem in four different models. These models differ in (i) whether or not the database T is private, (ii) who is the owner of T, and (iii) who may query T. We assume that the number of entries in T is N and, w.l.o.g., that each $t_i \in T$ is of length n. The result of a query q made to T, the approximate match, is denoted by $\mathrm{Match}(q, T)$. In each model, Bob possesses T. The models are the following.

- **Private Information Matching model (PIM)**. Alice has a query string q, and wants to know $\mathrm{Match}(q, T)$ without revealing q nor $\mathrm{Match}(q, T)$ to Bob. Bob, the *sole* possessor of T, doesn't want to reveal any $t_i \in T$ to Alice except what can be derived from $\mathrm{Match}(q, T)$. Alice has to query T through Bob.
- **Private Information Matching from Public Database model (PIMPD)**. As PIM, but T is public; the privacy concerns is that Alice doesn't want to reveal q nor $\mathrm{Match}(q, T)$ to Bob.
- **Secure Storage Outsourcing model (SSO)**: The owner of T is Alice, but T has been outsourced to Bob (e.g. for storage space reasons). Alice wants to query T without revealing T nor q to Bob.
- **Secure Storage and Computing Outsourcing model (SSCO)**: SSO with the following extension: any individual may query T, but Alice should be aware of any such queries. The individual making the query should learn the distance of the closest match from the query, while this should be kept secret from Alice.

Examples of situations which conform to one/some of the four models appear in e–commerce, transactions in certain large databases such as public health records, insurance records, banking records, and other liability related databases.

4 Overview of Results

In [1], the authors present protocols for approximate pattern matching in the SDA context for the following combinations of models and metrics.

For the PIM models protocols are given for

- $\sum_{i=1}^{n} (a_i - b_i)^2$,
- $\sum_{i=1}^{n} |a_i - b_i|$, and
- $\sum_{i=1}^{n} f(a_i, b_i)$ (general metrics).

The solutions dealing with the PIM model directly apply to the PIMPD model, as in PIMPD Bob may learn about entries in T. The author barely observe that there exists no secure two–party non–interactive solution to approximate pattern matching in context of the PIMPD model.

In addition, for the SSO and SSCO models protocols dealing with the $\sum_{i=1}^{n} (a_i - b_i)^2$ metric is given.

We discuss these two protocols in more depth in the following section.

Although the authors claim that they have a solution for PIM when using the string edit distance metric, they do not give any explicit hint on what kind of a protocol the solution is based on.

The presented protocols and the properties of the protocols (namely communications complexity (CC) and whether a third party is needed (3rd?)) are presented in the table below. In the table, n is the length of each string in T (as well as the length of the query string), N is the number of entries in T, m is the size of the alphabet the characters of which the strings consist of, and W is a parameter based on the number of random numbers used in the protocol, that affects the accuracy of the protocol in the PIM case with metric $\sum_{i=1}^{n} |a_i - b_i|$ (in this case the protocol is based on a Monte Carlo technique).

Model	**Metrics**	**CC**	**3rd ?**
PIM	$\sum_{i=1}^{n}(a_i - b_i)^2$	$\mathcal{O}(nN)*$	yes
	$\sum_{i=1}^{n} \vert a_i - b_i \vert$	$\mathcal{O}(nWN)*$	yes
	$\sum_{i=1}^{n} f(a_i, b_i)$	$\mathcal{O}(mnN)$ $**$	yes
SSO	$\sum_{i=1}^{n}(a_i - b_i)^2$	$\mathcal{O}(n)*$	no
SSCO	$\sum_{i=1}^{n}(a_i - b_i)^2$	$\mathcal{O}(n^2)*$	yes

It is unclear how communication complexity in those cases in which we have $*$ in the **CC** column in the table above does not appear to depend on the size of the alphabet. It should be reasonable to assume that in all of the cases, we should have something like $\log m$ dependency on the size of the alphabet. Thus it is not clear how and why in some cases the protocols could work on *infinite* alphabets. Moreover, for some reason in the case that is marked with $**$ a linear, i.e., not logarithmic, dependency on the size alphabet is claimed for some reason.

The common reason for the need of a third party in the protocols for PIM (and thus also for PIMPD) is that in the protocols, Alice and Bob need to compute the scalar product of two vectors $\mathbf{x}$ and $\mathbf{y}$ that are private knowledge of Alice and Bob, respectively, so that Bob doesn't learn $\mathbf{x}$ and Alice doesn't learn $\mathbf{y}$. The solution that is presented in [1] to this problem requires a third party.

It can be clearly seen, that the solutions given for PIM are not practical, as the communication complexity of the protocols depends on the number of entries in the database.

5 In-Depth Look at the Protocols

We now consider two of the protocols presented in [1]. We only discuss the practical protocols (for SSO and SSCO), as, apart from theoretical interest, they are interesting also for their low communication complexity. For more in-depth information on the other protocols presented, the interested reader is referred to [1].

5.1 Protocol for SSO

Recall that in this model the database T is Alice's, and she wants to outsource it to Bob. A trivial solution would be that Alice encrypts T before outsourcing. Every time Alice wants to query T, Bob sends the encrypted T to Alice. However, this would result in communication complexity linear to the number of entries in T, and thus the approach is unusable.

The solution proposed in [1], too, uses the idea of perturbing the contents of T before outsourcing. However, in this solution, Bob is the one conducting the matching, without learning anything about the query or the contents of T.

More formally, let $\mathbf{Q} \in \mathbb{Z}^{(n+3)\times(n+3)}$ be a random invertible matrix, and R, R_A and R_i, $i \in \{1, \ldots, N\}$, random numbers, all of which are private to Alice. For each string $t_i = t_{i,1} \ldots t_{i,n} \in T$, we have a vector $\mathbf{t}_i = (\sum_{k=1}^{n} t_{i,k}^2 + R - R_i, t_{i,1}, \ldots, t_{i,n}, 1, R_i)$. In T', the outsourced version of T, we have the entry $\mathbf{t}'_i = \mathbf{Q}\mathbf{t}_i{}^T$($\mathbf{t}^T$ is the transpose of $\mathbf{t}$ here).

Let $q = q_i \ldots q_n$ be Alice's query string. The protocol steps are as follows.

1. Alice
    - generates R_A,
    - constructs $\mathbf{q} = (1, -2q_1, \ldots, -2q_n, R_A, 1)$, and
    - sends $\mathbf{q}\mathbf{Q}^{-1}$ to Bob.
2. Bob
    - computes $\text{score}_i = \mathbf{q} \cdot \mathbf{t}_i{}^T$ for each $\mathbf{t}'_i \in T'$,

- determines $\arg\min_{i=i}^{N} \text{score}_i$, and
- sends $\mathbf{t}'_i$ to Alice.

3. Alice determines the closest match $\mathbf{t}_i = \mathbf{Q}^{-1}\mathbf{t}'_i$.

In step 2, Bob can compute $\mathbf{q} \cdot \mathbf{t}_i{}^T$ as

$$\begin{aligned}\mathbf{q} \cdot \mathbf{t}_i{}^T &= \mathbf{q}\mathbf{I}\mathbf{t}'^T_i = \mathbf{q}(\mathbf{Q}^{-1}\mathbf{Q})\mathbf{t}'^T_i \\ &= \mathbf{q}\mathbf{Q}^{-1} \cdot \mathbf{Q}\mathbf{t}_i{}^T = \mathbf{q}\mathbf{Q}^{-1} \cdot \mathbf{t}'^T_i.\end{aligned}$$

The scores can be computed by Bob, as we have that

$$\mathbf{q} \cdot \mathbf{t}_i{}^T = \sum_{k=1}^{n} t_{i,k}^2 - 2\sum_{k=1}^{n} q_k t_{i,k} + R + R_A,$$

and recalling the metric

$$\sum_{k=1}^{n}(q_k - t_{i,k})^2 = \sum_{k=1}^{n} t_{i,k}^2 - 2\sum_{k=1}^{n} q_k t_{i,k} + \sum_{k=1}^{n} q_k^2,$$

we get

$$\begin{aligned}\text{score}_i &= \mathbf{q} \cdot \mathbf{t}_i{}^T \\ &= \sum_{k=1}^{n} t_{i,k}^2 - 2\sum_{k=1}^{n} q_k t_{i,k} + R + R_A \\ &= \sum_{k=1}^{n}(q_k - t_{i,k})^2 + (R + R_A - \sum_{k=1}^{n} q_k^2),\end{aligned}$$

where the second term is constant w.r.t. the entries in T, and does thus affect the relative scores.

Disregarding the possible dependency on the size of the alphabet, the communication complexity of this protocol is $\mathcal{O}(n)$, that is, linear in the length of the strings, as the only communication between Alice and Bob consists of sending a string of length n two times.

5.2 Protocol for SSCO

In the SSCO, apart from Alice, any other individual, say Carl, may also query T through Alice, so that Alice only learns that someone has made a query. The solution differs from the SSO case, as now Carl and Alice need to communicate both before Carl may query Bob (as Alice has private $\mathbf{Q}$, and doesn't want to reveal it) and after Bob's response, to unravel the distance to the closest match. Indeed, Carl only learns the distance of his query to the closest match, not the closest match itself.

In this solution T is perturbed as follows. Let $\mathbf{Q} \in \mathbb{Z}^{(n+4)\times(n+4)}$ be a random invertible matrix, and R_A, R and R_i, $i \in \{1, \ldots, N\}$, random numbers, all of which are private to Alice. We also have a random number R_C private to Carl. For each string $t_i = t_{i,1} \ldots t_{i,n} \in T$, we have a vector $\mathbf{t}_i = (\sum_{k=1}^{n} t_{i,k}^2 + R - R_i, t_{i,1}, \ldots, t_{i,n}, 1, 1, R_i)$. In T', the outsourced version of T, we have the entry $\mathbf{t}'_i = \mathbf{Q}\mathbf{t}_i{}^T$.

Let $q = q_i \ldots q_n$ be Carl's query. The protocol is as follows.

1. Carl generates R_C.
2. Alice generates R_A and R.
3. Carl and Alice jointly compute $\mathbf{q} = R(1, -2q_1, \ldots, -2q_n, R_C, R_A, 1)\mathbf{Q}^{-1}$.
4. Carl sends $\mathbf{q}$ to Bob.
5. Bob
   - computes $\text{score}_i = \mathbf{q} \cdot \mathbf{t}_i{}^T$ for each $\mathbf{t}'_i \in T'$, and
   - sends $\text{score} = \min_{i=i}^{N} \text{score}_i$ to Alice.
6. Alice sends $\text{score}' = \frac{\text{score}}{R} - R_A$ to Carl.
7. Carl determines the distance to the closest match: $\text{distance} = \text{score}' + \sum_{k=1}^{n} q_k^2 - R_C$.

Again, Bob can compute the scores, as we have that

$$\begin{aligned}\text{score}_i &= \mathbf{q} \cdot \mathbf{t}_i{}^T \\ &= R(1, -2q_1, \ldots, -2q_n, R_C, R_A, 1)(\mathbf{Q}^{-1}\mathbf{Q})\mathbf{t}'_i \\ &= R(1, -2q_1, \ldots, -2q_n, R_C, R_A, 1)\mathbf{t}'_i,\end{aligned}$$

where R does not affect the relative scores. From this it follows that

score_i

$$= R\left[\sum_{k=1}^n t_{i,k}^2 - R_i - \sum_{k=1}^n 2t_{i,k}q_k + R_C + R_A + R_i\right]$$

$$= R\left[\sum_{k=1}^n t_{i,k}^2 - \sum_{k=1}^n 2t_{i,k}q_k + R_A + R_C\right]$$

$$= R\left[\sum_{k=1}^n (t_{i,k} - q_k)^2 + (R_A + R_C - \sum_{k=1}^n q_k^2)\right],$$

where as in the SSO case, the second term does not affect the relative score. Now we have

$$\text{score}'_i = \frac{\text{score}}{R} - R_A = \sum_{k=1}^n (t_{i,k} - q_k)^2 + (R_C - \sum_{k=1}^n q_k^2),$$

and, at last, Carl gets

$$\text{distance} = \text{score}' + \sum_{k=1}^n q_k^2 - R_C = \sum_{k=1}^n (t_{i,k} - q_k)^2.$$

As can be seen, this protocol is relatively close to the one in the SSO case. Here, R_A and R are used to having Carl learn only the closest score, and R_C to keeping the actual score secret from Alice. The communication complexity $\mathcal{O}(n^2)$ (disregarding the dependency on the size of the alphabet) comes from step 3, in which Carl and Alice jointly compute $\mathcal{O}(n)$ scalar products, each of which contributing $\mathcal{O}(n)$ with the protocol for computing scalar products presented in [1]. Notice that this is where a third (fourth, actually) party comes into play.

6 Conclusion

This paper is a survey of a paper by W. Du and M. J. Atallah [1] in which the problem of approximate pattern matching in the secure database access problem context is considered in four different models. The motivation in developing protocols for such specific problems is that in special cases the communication complexity of a protocol can be greatly decreased, making the protocols practical. Some of the proposed protocols are still on a theoretical level, communication complexity wise, while in two cases complexity-wise practical solutions are presented. Solutions to some cases are totally left out of discussion, oddly. Of the two cases, in the other an additional party is required to establish the privacy objectives. It should be noticed that the claimed independency on the size of the alphabet considered in some of the solution seems to be inconsistent. It would have been nice to see more detailed discussion on the solutions in [1] (especially for the cases that were totally dismissed), as the introductory phase of the paper could have maybe been written in a more compact way.

References

[1] W. Du, M.J. Atallah. Protocols for Secure Remote Database Access with Approximate Matching. In 7th ACM Conference on Computer and Communications Security (ACM-CCS 2000), The First Workshop on Security and Privacy in E–Commerce, Athens, Greece, November 1–4 2000.

[2] A. Apostolico, Z. Galil (ed). Pattern Matching Algorithms. Oxford University Press, 1997.

[3] A. Yao. Protocols for Secure Computations. In Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982.

[4] B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan. Private Information Retrieval. In Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science, 1995