Non-interactive fuzzy private matching

Abstract. Two fuzzy private matching protocols are introduced to allow a client to securely compare a list of words to a server list, and discover only those words on the server list that are similar to his, while the server learns nothing. The first protocol achieves perfect client security, while the second achieves almost-privacy and perfect server security. Both protocols are efficient in both communication and computation complexity: for lists of length n, only $O(n)$ communication and $O(n^2)$ computation is needed.

1 Introduction

In our current information society, large amounts of data are being accumulated. Naturally, there is interaction between parties having different data. A common problem is that two parties want to find out whether they share certain data. However, neither is really willing to reveal all his data to the other. For example, one can think of cooperative data mining, where two competitors want to cooperate in discovering market trends from their sales data - without revealing that data to the competitor. Other examples include online dating, peer to peer services, recommendation services, electronic health records, and so on. In practice, this problem is often solved by using a trusted third party, or by one party sending (part of) his data first regardless of any objections.

Both parties, called the client and the server, will be assumed to have a list of words. The problem now is to design a secure protocol, by which the client learns which of his words are also on the servers list, but which does not provide the server with any information about the clients list. Private matching was introduced to solve this problem securely without the use of a third party [1]. For practical cases, one often wishes to find out not only which words match exactly, but also the near matches - think of situations like misspellings and tastes which are never exactly equal. This problem is called fuzzy private matching, and it was left open by Freedman et al. [1] to design an efficient protocol that achieves this. This work will present two efficient protocols that achieve this goal in the semi-honest model.

2 Background

The root of the considered problem lies in private equality testing [2,3,4], or PET, where the client and the server each hold a single word and want to securely decide whether their words are equal. Assuming a word consisting of T symbols, the solution runs in complexity $O(T)$.

Freedman et al. [1] consider lists instead of single words. They develop protocols for private matching. Another natural direction to extend the problem is to look at similarity instead of equality - what if only t out of T symbols (“letters”) of a word have to be equal? Private approximation of functions was formalized by

Feigenbaum et al. [5]. In particular, they left open the question whether there are efficient protocols achieving private similarity testing. Indyk and Woodruff [6] answer this question positively.
Combining both these extensions, one arrives at fuzzy private matching. Freedman et al. gave a first protocol with communication complexity $O(n\binom{T}{t})$, and posed the question whether the same could be achieved without the $\binom{T}{t}$ factor. Indyk and Woodruff [6] give a construction with $\tilde{O}(nT + n^2)$ communication. Chmielewski and Hoepman [7] achieve $O(nT)$ communication by moving the binomial factor to the computation complexity, which is $O(n^2\binom{T}{t})$ in their best protocol.
Here two solutions will be presented which eliminate the binomial factor, and have communication complexity $O(nT)$, and computation complexity $O(n^2)$.

3 Preliminaries

3.1 Notation and problem definition

Assume that the words x_i to be matched have length T and symbols from a field $\mathbb{F}$: $x_i \in \mathbb{F}^T$. The server will have a list $L_S = \{y_1, \dots, y_{n_S}\}$ of n_S words, while the clients list $L_C = \{x_1, \dots, x_{n_C}\}$ consists of n_C words. The goal is to design a protocol that allows the client to discover words on the server list that are similar to his. To be more precise, the client should learn the set

$$\mathcal{R} = \{y_i \mid \exists j, d(y_i, x_j) \leq T - t\},$$

where d denotes Hamming distance (i.e. t out of T symbols of the word should match).

Definition 1. *A t-out-of-T (h_C, h_S) fuzzy private matching scheme is a protocol between the client and the server which outputs the set $\mathcal{R}$ of those server words to the client, which match on at least t symbols with one of the clients words. For each (non-matched) server word, the entropy loss is at most h_S, and for each client word the entropy loss is at most h_C.*

Here, entropy loss is measured as $H_\infty(W) - \tilde{H}_\infty(W|\mathrm{FPM})$, where W is the distributions of words, and $\tilde{H}_\infty$ is the average min-entropy [8] left after the protocol FPM has run.
Note that the privacy requirements are somewhat relaxed - in the literature [1,6,7] only $(0,0)$ fuzzy private matching schemes are considered.

3.2 Secure sketches

Secure sketches were introduced as a kind of secure error-correcting code [8]. In particular, a secure sketch produces public information about its input w that does not reveal w, and yet it allows exact recovery of w given a value that is close to w. To be more precise we quote Dodis et al.:

Definition 2 ([8]). *An $(M, m, \tilde{m}, t)$-secure sketch is a pair of randomized procedures, "Sketch" (SS) and "Recover" (Rec), with the following properties:*
1. *The sketching procedure on input $w \in M$ returns a bit string $s \in \{0,1\}^\star$.*

2. *The recovery procedure takes an element $w' \in M$ and a bit string $s \in \{0,1\}^\star$. The correctness property of the secure sketch guarantees that if $d(w, w') \leq t$, then $Rec(w', SS(w)) = w$.*
3. *The security property guarantees that $\tilde{H}_\infty(W|SS(W)) \geq \tilde{m}$ for any distribution W over M with min-entropy m. In other words, an adversary who observes s can recover the input w with a probability at most $2^{-\tilde{m}}$.*

Note that this definition is adapted for Hamming distance d, and Dodis et al. also considered secure sketches for other similarity metrics. Here we will focus on the Hamming distance constructions, since this most closely matches our problem.

Constructing a secure sketch from MDS codes. As shown in the literature [9,8], error-correcting codes can be used to construct secure sketches. Similar to those constructions, an $[n, k, d]$ MDS code $\mathcal{C}$ (so $k = n - d + 1$) can be used to construct a $(\mathbb{F}^k, m, m - (n-k)f, \frac{d-1}{2})$ secure sketch as follows. Given $w \in \mathbb{F}^k$, calculate the unique tail e such that $(w||e) \in \mathcal{C}$ (here $||$ stands for concatenation). Since any k symbols in an MDS code can be used as the information symbols [10], such an e always exists and is unique. Recovery is done by decoding $(w'||e)$, and will succeed if $d(w||e, w'||e) = d(w, w') \leq \frac{d-1}{2}$. In particular, if we use a (generalised) Reed-Solomon code, for which efficient decoding algorithms are known (e.g. using the Berlekamp-Massey algorithm), we arrive at an efficient and deterministic sketch with small output size.

3.3 Adversary model

In this work, we assume a semi-honest adversary [11]. Informally, this means that both parties are expected to follow the protocol. As a measure for the security (both of the client and the server), we will take the entropy loss [8] defined as $h = H_\infty(W) - \tilde{H}_\infty(W|\mathrm{FPM})$, which measures how much information an eavesdropping adversary gains after the protocol FPM is run. Note that this information need not be deterministic. For instance, in the secure sketch construction above, the entropy loss is $(n-k)\log_2 q$. This means that out of q^k, there are still q^{2k-n} possibilities left for w, since there are exactly that many codewords ending in the given tail e.

4 Non-interactive Fuzzy Private Matching

In this section, a low-complexity - both in terms of communication and computation - protocol for private fuzzy matching will be introduced, based on secure sketches. There will be both a basic version, achieving perfect client security, as well as an interactive version, which achieves perfect server security.

4.1 Basic Protocol

The basic non-interactive protocol consists of three phases. Firstly, the server performs the setup. Then the server publishes information about his list. lastly, the client can use this information to discover which of his words are similar.

Setup Phase. In the setup phase, the server fixes an $[n, k, d]$ q-ary MDS code $\mathcal{C}$ with $n = 3T - 2t$, $k = T$ and $d = 2T - 2t + 1$. The server also publishes a decoding algorithm D for $\mathcal{C}$ which can efficiently correct $\frac{d-1}{2} = T - t$ errors. Note that such an efficiently decodable code exists [10], as long as q is chosen such that $n \leq q$ (take for example a generalized Reed-Solomon code).
Also, the server fixes an appropiate, publicly known, semantically secure encryption scheme (Enc,Dec).
Server operations. For each word y_i in the server list L_S, the server finds e_i such that $c_i = y_i||e_i \in \mathcal{C}$ (here $||$ stands for concatenation). Since y_i consists of $k = T$ coordinates and $\mathcal{C}$ is an MDS code, there is a unique codeword that starts with these coordinates. Let $k_i = \mathrm{D}(c_i)$. The server publishes the pair $(e_i, \mathrm{Enc}_{k_i}(y_i))$. All in all, the server publishes a list of pairs $\{(e_i, \mathrm{Enc}_{k_i}(y_i))|1 \leq i \leq n_S\}$.
Client operations. In order to find out whether a word x_i on the client list L_C matches a published pair (e, c), the client performs the following operations. Firstly, the client applies the decoding algorithm D to obtain $k = \mathrm{D}(x_i||e)$. If the decoding algorithm fails, x_i doesn't match that server word. Otherwise, the client continues by trying to decrypt c with the obtained k to find $y = \mathrm{Dec}_k(c)$. If the decryption is succesful, the word x_i matches y on the server list, and the client has retireved y.
The client will try to match each pair of the published server list to words in his own list L_C, until he either finds a succesful match or he has exhausted his own list.

4.2 Protocol analysis

In this section, the correctness, communication and computational complexity, and the security of the basic protocol will be discussed. The basic protocol will turn out to be a t-out-of-T $(0, 2f(T - t))$ fuzzy private matching scheme.

Correctness. First of all, assume the client has a word x matching the server pair $(e_i, \mathrm{Enc}_{k_i}(y_i))$. Since x matches the pair, the following Hamming weight can be bounded $d(x||e_i, y_i||e_i) = d(x, y_i) \leq T - t$, and thus D is able to efficiently decode and reconstruct k_i. Using this key, the client is able to find the original server word y_i.
On the other hand, if the server pair doesn't match the client word, we have that $d(x||e_i, y_i||e_i) > T - t$, which means that one will not find k_i using D. The decoding algorithm will either abort, or return a key which is of no use to decrypt y_i.

Complexity. The communication complexity of this protocol is only $n_S(3T - 2t)$ symbols. This is the same order of magnitude as the best known results from the literature [7]. However, the clients time complexity is only $O(n_S n_C)$, which substantially improves upon the time complexity of $O(n_S n_C \binom{T}{t})$ [7]. The servers time complexity is $O(n_S)$.

Security. Since the basic protocol is a non-interactive scheme, the client list remains perfectly secure - the server never gets any information about it.

To analyze the security of the server list, we first remark that for each word, the first part published by the server is the secure sketch discussed in Section 3.2. In fact, this is an $(\mathbb{F}^T, T(1-\delta)f, 2tf - T(1+\delta)f, T-t)$ secure sketch [8, Theorem 5.1] for any $0 \leq \delta < 1$, where $f = \log_2(q)$. To show this claim, assume that the words on the server list are drawn from a distribution with min-entropy $H_\infty(Y) = T(1-\delta)f$. Then the average min-entropy given the sketch E is $\tilde{H}_\infty(Y|E) = 2tf - T(1+\delta)f$, since $\mathcal{C}$ is a q-ary MDS code.
This shows that publishing e inevitably leaks information about y. In particular, we need $t > (1+\delta)\frac{T}{2}$ in order to have any min-entropy left. Hence, this approach is best suited for fuzzy private matching with $t \approx T$.

4.3 Interactive Protocol

The basic protocol achieves perfect client security. However, the server security is only computational. In fact, the server essentially publishes $n - k = 2T - 2t$ symbols of each of his words. Also, the server has no control over the length of the client list - which allows the client to perform a brute-force attack on the published server list.
In order to remedy these problems, the protocol can be reversed. Pick a suitable collision-resistant hash function h. For each word in the client list, the following protocol is executed:

1. Assume the client word is x. The client calculates e such that $c = (x||e) \in \mathcal{C}$. For the same reasons as in the basic protocol, there is a unique codeword that starts with these coordinates. Let $k = \mathrm{D}(c)$. The client sends the pair $(e, h(k))$ to the server.
2. For each word y_i in the server list, the server decodes $y_i||e$ using the efficient decoding algorithm D. If decoding succeeds, the server calculates $h(\mathrm{D}(y_i||e))$ and compares it to the hash value sent by the client. If these are equal, the server word matches the client word, and the server sends y_i to the client, encrypted with the users key.

This interactive protocol is perfectly secure for the server. However, this comes at a price. First of all, now the clients security is only computational - by the same arguments, the client publishes information about his words (in fact, the entropy loss is equal to the non-interactive case). Moreover, since the server now has to do the decoding, the computational complexity shifts to the server, which may be undesirable. Lastly, perhaps most importantly, this protocol does not achieve the exact goals we set out at the start - the server will learn which of its words match the clients list. Unfortunately, this seems to be unavoidable when perfect server security is desired with small communication complexity (i.e. less than $n_C n_S$), since the server can always run his part of the protocol twice (or more) with adjusted server lists - it seems there is not enough communication possible for the client to exert control over how often his data is used. Thus the interactive protocol is a t-out-of-T $(2f(T-t), 0)$ fuzzy almost-private matching scheme.

4.4 Protocol enhancements

In this section some enhancements to the protocol will be introduced. While they are written from the perspective of the basic protocol, the enhancements apply to the interactive version as well.

Keylength. The basic protocol assumes an encryption scheme with a keylength of Tf bits. In particular, this means that T must be large enough to not allow for a brute-force attack on $\mathrm{Enc}_{k_i}(y_i)$. Moreover, in practice, such a scheme might be hard to find, especially if one wants to conform to standards (like AES).
Assume that one wants to use an encryption scheme which has a keylength of $l = (T + \tau)f$ bits. For simplicity we will consider the case where f is a divisor of l (in the general case, if we discard a few bits the extra entropy loss will be small). To arrive at an l-bit key, the server should select an $[3T - 2t + \tau, T + \tau, 2T - 2t + 1]$ MDS code $\mathcal{C}$ in the setup phase. Then for each server word y_i the server generates a string t_i of τ random symbols, and constructs the codeword $c_i = (y_i||t_i||e_i)$. Instead of publishing e_i, the server will publish $(t_i||e_i)$. Note that this construction does not modify the resulting min-entropy of the protocol - it allows one to use a standard encryption scheme, and prevents brute-force attacks when T is too small.

Introducing user dependence. The biggest weakness of the non-interactive protocol is that the server has no control over the client. In particular, the client (or anyone who has access to the published server list) can keep trying to find a match until he can successfully decrypt the server word. In order to limit this somewhat, the server may choose to adapt the protocol slightly and encrypt each e_i with the users key (for this we have to assume either a public key, or a shared secret key). This protects the server against eavesdropping on the list – the server inherently has no control over the clients list.

Using list decoding. Recent developments in coding theory [12,13] allow one to (list) decode beyond the traditional $\frac{d-1}{2}$ bound. For (generalised) Reed-Solomon codes, it is possible to decode up to $e = n - \sqrt{kn}$ errors (which is strictly larger than $\frac{d-1}{2}$) in about $\tilde{O}(n^{12})$ time [13, Theorem 6.16]. It is still an open problem whether this is the best possible result for specific codes, like the generalized Reed-Solomon codes used here.
These results could be used by a client in the basic protocol to correct more errors than anticipated, resulting in the ability to match words that are (slightly) further apart. However, we can also use these results positively: if we can efficiently perform Sudan's algorithm [12], the protocol only needs a code $\mathcal{C}$' that can list decode up to $e = T - t$ errors. This will lead to a smaller length of the used code.

5 Conclusions

Two protocols for t-out-of-T fuzzy private matching were constructed using secure sketches based on MDS codes for $2t > T$. Both run in small communication complexity, as well as low computation complexity. The first, basic protocol is a $(0, 2f(T - t))$ fuzzy private matching scheme, or in other words it achieves perfect client security, and computational (exponential in t) server security in a non-interactive fashion. The second, interactive protocol enjoys perfect server security and computational client security, i.e. it is a $(2f(T - t), 0)$ fuzzy almost-private matching scheme. However, in the last scheme the server learns which words from his list are similar to the clients list.

References

1. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: EUROCRYPT. (2004) 1–19
2. Fagin, R., Naor, M., Winkler, P.: Comparing information without leaking it. Communications of the ACM **39** (1996) 77–85
3. Boudot, F., Schoenmakers, B., Traore, J.: A fair and efficient solution to the socialist millionaires' problem. Discrete Applied Mathematics **111** (2001) 23–36
4. Lipmaa, H.: Verifiable homomorphic oblivious transfer and private equality test. In: ASIACRYPT. (2003) 416–433
5. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M.J., Wright, R.N.: Secure multiparty computation of approximations. Lecture Notes in Computer Science **2076** (2001) 927+
6. Indyk, P., Woodruff, D.P.: Polylogarithmic private approximations and efficient matching. In Halevi, S., Rabin, T., eds.: TCC. Volume 3876 of Lecture Notes in Computer Science., Springer (2006) 245–264
7. Chmielewski, L., Hoepman, J.H.: Fuzzy private matching. Submitted for publication (2007)
8. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: EUROCRYPT. (2004) 523–540
9. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: ACM Conference on Computer and Communications Security. (1999) 28–36
10. MacWilliams, F., Sloane, N.: The theory of error-correcting codes. North-Holland Publishing Co., Amsterdam (1977) North-Holland Mathematical Library, Vol. 16.
11. Goldreich, O.: Secure multi-party computation. Working Draft (2000)
12. Sudan, M.: Decoding of Reed Solomon codes beyond the error-correction bound. J. Complex. **13** (1997) 180–193
13. Guruswami, V.: List Decoding of Error-Correcting Codes (Winning Thesis of the 2002 ACM Doctoral Dissertation Competition). Volume 3282 of Lecture Notes in Computer Science. Springer (2004)