Manuscript Details

Manuscript number	COSE_2018_560_R1
Title	Communication-efficient Private Distance Calculation Based on Oblivious Transfer Extensions
Article type	Full Length Article

Abstract

We propose a general framework for computing privacy-preserving distance metrics (PPDM) in the two-party setting in order to improve communication complexity by benefiting from 1-out-of-n oblivious transfers. We implement privacy-preserving Euclidean distance, Cosine similarity and Edit distance protocols while the PPDM framework is easily extendable to address other distance measures. These protocols have direct applications in privacy-preserving one-to-many biometric identification in which two parties known as client and server want to find the best match between their inputs. The client's input is compared to all the records in the server's database. Our threat model is semi-honest adversaries. We extensively evaluate our PPDM framework. And, we theoretically show the improvement of PPDM over related work.

Keywords	Private Distance Calculation; Oblivious Transfer; Distance-based Similarity; Efficiency; Communication Complexity
Taxonomy	Cryptography, Database Security, Web Services, Network Security
Corresponding Author	parisa kaghazgaran
Corresponding Author's Institution	Texas A&M University
Order of Authors	parisa kaghazgaran, Hassan Takabi, Flannery Hope Currin, Armando Soriano
Suggested reviewers	Murat kantarcioglu

Submission Files Included in this PDF

File Name [File Type]

Cover Letter.docx [Cover Letter]

response_letter.pdf [Response to Reviewers]

elsarticle-template.pdf [Manuscript File]

Biography.docx [Author Biography]

Submission Files Not Included in this PDF

File Name [File Type]

Elsevior-Computer&Security.zip [LaTeX Source File]

To view all the submission files, including those not included in the PDF, click on the manuscript title on your EVISE Homepage, then click 'Download zip file'.

Research Data Related to this Submission

There are no linked research data sets for this submission. The following reason is given:
Synthetic Data has been used and we publish our source code wherein data is generated within a Java method.

Communication-efficient Private Distance Calculation Based on Oblivious Transfer Extensions

Authors:

1. Parisa Kaghazgaran, PhD student, Texas A&M University, College Station, TX, US
2. Hassan Takabi, Assistant Professor, University of North Texas, Denton, Texas, US
3. Flannery Hope Currin, Research Intern, University of North Texas, Denton, Texas, US
4. Armando Soriano, Research Intern, University of North Texas, Denton, Texas, US

Statement on the Revision of COSE_2018_560 paper, entitled "Communication-efficient Private Distance Calculation Based on Oblivious Transfer Extensions "

We would like to thank all the reviewers, Associate Editor, and Editor in Chief for their valuable feedback. We have revised our original submission to incorporate the many helpful suggestions of the reviewers. In particular our revisions address following points:

- **Reviewer 1**

This paper proposes a novel 2-party PPDM solution. The focus of this method is to use 1-out-of-N OT strategy to reduce the communication overheads while the distance computing, which enables the usage of the secured distance metric calculating in large-scale biometric applications. I think the idea of combing hexadecimal encoding and 1-out-of-16 OT together for minimizing information overlapping between the two game players (so as to minimize the information leakage) is interesting and promising. My minor concerns are as follows:

1. For the secured edit distance, a threshold is required to decide whether the execution will be stopped (see line 330), how is the threshold chosen? Is it chosen by hand or an empirical criterion guiding the choice? I would guess there is a trade-off depending on the threshold: the early stop of the execution might reduce the computational cost, while it might increase the probability of being biased from the true distance. In that case, how do we measure the impact of the threshold choice, over the performance of the distance calculation?

Response:
Empirically, we set the value of *k* to be 60, 80 and 100. Since the goal of privacy preserving Edit distance in genome related scenarios is to return the top most similar pattern, by exploring the dataset provided by iDASH security and privacy workshop, we found that by setting the threshold to be in this range the algorithm will return reasonable number of similar (about 10) patterns. However, in our implementation value of threshold is an input parameter that can be varied based on the scenario. Also, the average length of genome patterns is 3,500 characters so that a threshold between 60 to 100 will return the most similar patterns i.e., patterns which only need 60 to 100 transformations to replicate each other. On the other hand, for shorter patterns the threshold should be a smaller value for a good trade-off between accuracy and information leakage.

how do we measure the impact of the threshold choice, over the performance of the distance calculation?

Response:
The value of threshold impacts communication complexity linearly. In our experiments, we evaluate the privacy preserving Edit distance algorithms with different values of threshold as 60, 80 and 100. In section 4.2, “Secure Comparison→ communication complexity”, we discuss the impact of threshold on performance in detail. However, execution time does not increase linearly. For example, the protocol takes 45 second to be executed with threshed=60 while with the same configuration it only takes 55 second when threshold is 100.

2. For biometric applications, the scalability is usually one of the most important factors measuring usability of the recognition service. In the experimental study, I notice that the running time costs of the PPDM method with respect to increasingly larger datasets are given. My questions would be

1) Does the execution time include the communication time cost?

Response:
Yes, the reported execution time in the paper includes both computation and communication time as the whole. Basically, the reported execution time measures the whole time spent to read the data, perform the computation and transfer messages between parties.

2) It is interesting to see how the method scales on an even larger dataset. For example, we can image the number of feature vectors contained in the gallery dataset increase by a factor of 10, like 1000,10000 and 100000, in order to see if the method scales with a linear, super-linear or even quadratic rate.

Response:
The execution time does not scale proportionally when we double up the size of the database. For example, the execution time is 3 sec and 3.2 sec in the presence of databases with 128 and 320 in size respectively when message length is 8-bit. In our implementation that is now public[1], the size of the database is an input parameter so that can the database size can be set with any arbitrary number. To keep the consistency through the paper, we report the results on databases with size of 128, 256, 320 and 512 (number of samples).

[1]http://people.tamu.edu/~kaghazgaran/Privacy_Preserving_Distance_Calculation_Framework.7z

3) Is there any consideration supporting the choice of the given range of the symmetric and public-key parameter (see line 85 and all over the experiments)?

Response:
We evaluate our approach with symmetric security parameter to be either 80 or 128 and public security parameter to be 1024 or 2048 that reflect short-term (80 and 1024) and log-term/ stronger security (128 and 2048) parameters with respect to NIST recommendations.

- **Reviewer 2**

This paper proposed a private identification scheme based on oblivious transfer, or more specifically, privacy-preserving distance calculation based on oblivious transfer. The problems with the submission are as follows:

1. The title of the paper is not accurate in that the key contribution of this paper is not identification but distance/similarity calculation. Private identification is an application of distance/similarity calculation.

Response:
The title has been changed to “Communication-efficient Private Distance Calculation Based on Oblivious Transfer Extensions ” To reflect the main contribution of the paper.

2. The key technique used in this paper is a combination of oblivious transfer. Although the proposed scheme does have some improvements over existing ones, the improvements seem limited according to Table 1 (16% improvements over Algorithm 3).

Response:
Authors do believe 16% improvement is a significant improvement in cryptographic protocols. However, in the presence of messages with short bit-length e.g., 8-bit we achieve 90% improvement over Homomorphic encryption-based protocols (Table 1).

3. The organization of the paper is difficult for readers to follow. It is hard to tell whether Section 5 discusses experiments or privacy-preserving distance measurement. Moreover, literature reviews have been discussed for multiple times in Section 5.

Response:
The structure has been changed in this way:
Also, all related work is put together in a separate Section (Section 7).

4. Experiments did not show difference between the proposed schemes with existing schemes.

Response:
Table 1 compares the communication complexity of Arithmetic multiplication protocols described in Algorithms 2 to 4 with respect to different values of symmetric and public key security parameters and different lengths of messages in bit for one multiplication. As it can be seen from the "Improvement" rows, our protocol outperforms two other approaches significantly, in particular for short messages like 8-bit messages. For example, in the presence of messages with 8-bit length our approach has about 90% improvement over Homomorphic based multiplication and about 50% improvement over 1-out-of-2 OT based multiplication. By increasing length of messages, we can see as we choose stronger security parameters, our protocol outperforms its peers.

Also, our main contribution is improvement of bandwidth complexity that table 1 confirms the success of our protocol compared to its peers in this regard.

However, for Edit distance protocol the state of the art work is run on virtual machines available from iDASH security and privacy workshop. We also run our protocol on the same instances to replicate the same execution environment. We add an analysis section in the paper to compare our protocol with related work (page 28→ Analysis)

Code :
http://people.tamu.edu/~kaghazgaran/Privacy_Preserving_Distance_Calculation_Framework.7z

Communication-efficient Private Distance Calculation Based on Oblivious Transfer Extensions

Parisa Kaghazgaran[1], Hassan Takabi[2], Flannery Hope Currin[2], Armando Soriano[2]

[1]*Texas A& M University,* [2]*University of North Texas*

Abstract

We propose a general framework for computing privacy-preserving distance metrics (PPDM) in the two-party setting in order to improve communication complexity by benefiting from 1-out-of-n oblivious transfers. We implement privacy-preserving Euclidean distance, Cosine similarity and Edit distance protocols while the PPDM framework is easily extendable to address other distance measures. These protocols have direct applications in privacy-preserving one-to-many biometric identification in which two parties known as client and server want to find the best match between their inputs. The client's input is compared to all the records in the server's database. Our threat model is semi-honest adversaries. We extensively evaluate our PPDM framework. And, we theoretically show the improvement of PPDM over related work.

Keywords: Private Identification, Oblivious Transfer, Distance-based Similarity, Efficiency, Communication Complexity

1. Introduction

Secure Two-party Computation, introduced by Yao [1] and Goldreich-Micali-Wigderson (GMW) [2], solves problems in which two parties jointly want to evaluate a function on their private inputs without revealing any information except what can be inferred from the final results. A common function is distance calculation which has direct application to privacy-preserving biometric identification [3, 4, 5, 6, 7, 8]. In this scenario, a client holds a biometric pattern and a server holds a database of biometric patterns; they want to determine whether there is a match for the client's input in the server's database. However, the client does not want to reveal its pattern to the server, since it would enable the server to track the pattern's owner. For the similar reasons, the server will not disclose any information about its database to the client.

Privacy-preserving distance calculation has been studied extensively in recent years. Early protocols were based on pure (additively) Homomorphic Encryption (HE) techniques, e.g., [5]. Later work showed that protocols using generic secure computation techniques such as Yao's garbled circuits and GMW circuits outperform HE techniques. These protocols are based on either a combination of HE and circuit-based approaches [3, 4, 7, 6] or pure circuit-based techniques [9, 10, 11].

While it has been shown [12, 13, 14] that Oblivious Transfer (OT) is the fastest secure technique in a two party setting, particularly in privacy-preserving biometric identification [15, 16, 17], communication bandwidth and multiple communication rounds are the main bottleneck of OT-based protocols. For example, Schneider showed that an increase in the bit-length of inputs and/ or an increase in the number of the required OTs will lead to a significant increase in communication bandwidth [18].

A few recent work has improved the complexity of OT-based protocols [12, 13] and evaluate their applications on privacy-preserving biometric identification problem [14]. In this paper, we build upon existing protocols to improve communication complexity in OT-based protocols by using 1-out-of-n OTs. It should be noted that our work is based on pure OT and we aim to improve communication complexity in corresponding protocols.

In a related direction, Bringer et al. [16] proposed a protocol for privacy-preserving Hamming distance using OT. Then, Bringer et al. [17] generalized this protocol to enable privacy-preserving computation of other distance metrics using OT. Both work are built on binary representation of the inputs and take advantage of 1-out-of-2 OT. In contrast, in this paper, we propose a general framework applicable to any distance measure using 1-out-of-n OT considering hexadecimal representation of input data in order to improve communication complexity. The other main difference is that we address the secure Cosine similarity and Edit distance using ***pure OT*** for the first time in addition to classic privacy-preserving distance measures like Euclidean distance.

We skip Hamming distance since it deals exclusively with binary inputs and has been addressed many times using different techniques such as Homomorphic encryption [19, 20], Garbled circuits [10] and Oblivious Transfer [16].

1.1. Formal Definition

As shown in Figure 1, privacy-preserving biometric identification is accomplished in two phases: distance calculation and comparison.

The server and client want to privately compute the distance between one versus N feature vectors. The client's input represents a feature vector X while the server holds N feature vectors $Y_1, ..., Y_N$. Then, they jointly calculate the distance between X and each Y_i. At the end of the protocol, each party has an additive share of the distance. Suppose D indicates the actual distance, the client obtains D^0 and the server obtains D^1 such that $D^0 + D^1 = D$. The distance shares are then fed to a privacy-preserving comparison protocol to find the best match.

1.2. Our Contributions

We break our contributions as follows:

- We build a general framework for privacy-preserving distance calculation utilizing recent optimization of OT. This framework allows efficient calculation of various distance measures such as Euclidean, Manhattan and Mahalanobis distances, Scalar product, Cosine similarity and Edit distance.

- We improve the communication complexity by a factor of 2 by proposing a new protocol for secure Arithmetic multiplication using 1-out-of-16 OT

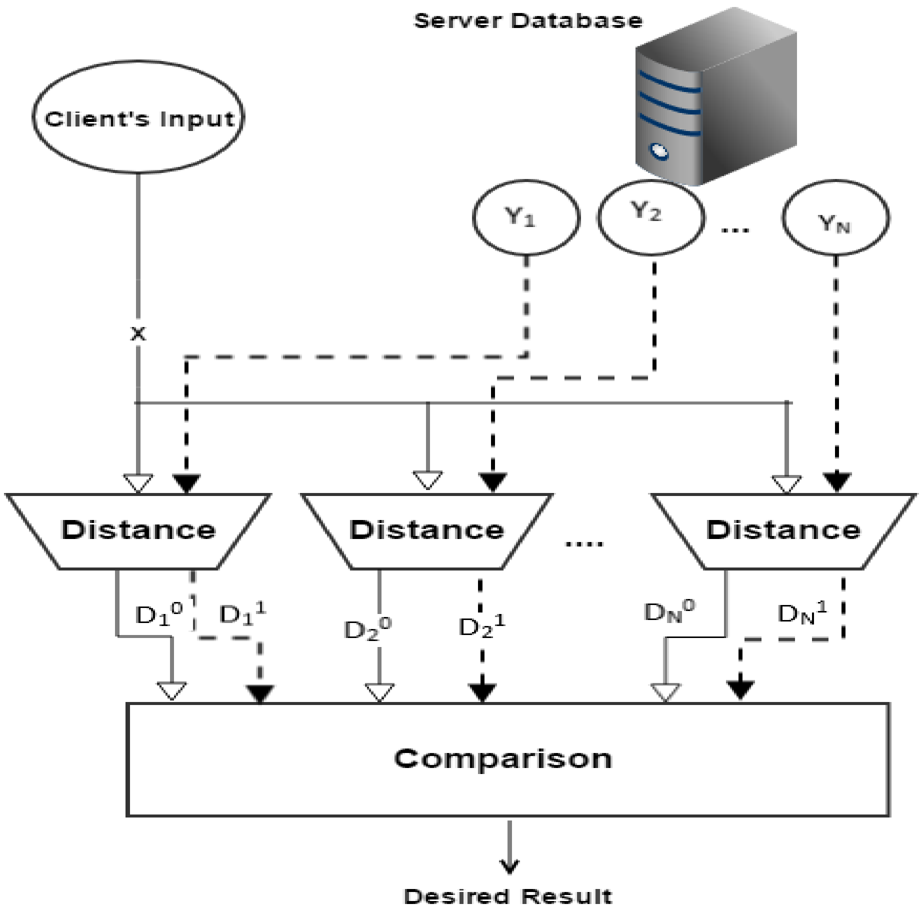

Figure 1: Identification Scenario

rather than 1-out-of-2 OT [21, 14] by leveraging hexadecimal representation of input data. As a result, our protocol also improves the computation complexity by reducing the number of required OTs.

- We propose efficient protocols for privacy-preserving Cosine similarity and Edit distance using OT for the first time. We present a novel protocol for secure comparison based on OT in order to address privacy-preserving Edit distance. We also analyze the security and accuracy of our framework.

- We implement the complete framework in Java and the source code is available from [1].

1.3. Notation

P_0 and P_1 refer to the client and server respectively. n denotes number of messages in 1-out-of-n Oblivious Transfer while r represents the selection bit. m is the size of the feature vectors also indicates number of required OTs in Algorithm 1. N indicates the size of the database. In the Edit distance algorithm, k refers to the distance threshold. D denotes the distance between two feature vectors. With respect to the shares of D, D^0 indicates the share belonging to P_0 and D^1 indicates the share belonging to P_1. l is the message bit-length in Oblivious Transfer. 2^l is the Arithmetic module.

κ and φ denote symmetric and public-key security parameters respectively. We evaluate our approach with $\kappa \in \{80, 128\}$ and $\varphi \in \{1024, 2048\}$ which reflect short-term (80 and 1024) and log-term (128 and 2048) security parameters with respect to NIST recommendations [22].

2. Background Information

In this section, we provide background information on privacy-preserving techniques. It should be noted that our main tool is Oblivious Transfer (OT) with Arithmetic Sharing. We first give a brief description of Homomorphic Encryption that is the basis of most previous work.

2.1. Homomorphic Encryption (HE)

In a public-key cryptosystem, the encryption of a given value x using public key pk is written as $[x]_{pk}$, or simply as $[x]$. An encryption is *additively Homomorphic* if one can compute $[x + y]$ from $[x]$ and $[y]$ without knowing the decryption key. Additionally, given $[x]$ and a constant integer c, one can compute $[c.x]$.

There are many public-key cryptosystems that satisfy this condition. However, related works mostly used the Paillier cryptosystem [23]. In this scheme,

[1] $http://people.tamu.edu/\sim kaghazgaran/Privacy\_Preserving\_Distance\_Calculation\_Framework.7z$

public key is the product of two private prime numbers, p and q, and φ denotes its bit-length where bit-length of the cipher-text is $2 \times \varphi$.

2.2. *Oblivious Transfer (OT)*

Two parties, called sender and receiver, participate in OT protocols. In 1-out-of-2 OT, the sender has two messages (x_0, x_1) and the receiver has a selection bit $r \in \{0, 1\}$. At the end of the protocol, the receiver only learns x_r and learns no information about x_{1-r} and the sender learns nothing about r. 1-out-of-2 OT can be generalized to 1-out-of-n OT in which the sender has n messages $\{x_0, ..., x_{n-1}\}$ and the receiver has a selection value $r \in \{0, ..., n-1\}$ to obtain x_r.

Preliminary OT-based protocols consist of expensive public-key operations while recent improvements of OT, called OT-extension [12, 13], allow the extension of a small number of base OTs using only symmetric operations. For the base OTs, a constant number (κ) of public-key operations is required.

To perform base OTs, two approaches are described in the following. The first approach is based on HE while the second one uses Deffie-Hellman (DH) key exchange protocol [24].

1-out-of-2 OT using HE: The receiver generates homomorphic encryption of $E(1-r)$ and $E(r)$ and sends them to the sender. The sender computes $E((1-r)x_0 + r.x_1)$ homomorphically without being able to decrypt the ciphertexts and sends the output to the receiver. The receiver decrypts the message to obtain $(1-r)x_0 + r.x_1$, which is equal to the desired x_r. Clearly, if $r = 0$ then x_0 will be obtained otherwise ($r = 1$), x_1 will be received.

1-out-of-2 OT using DH: The sender selects a random number $a \in Z_p$ and sends $A = g^a$ to the receiver (g is the group generator). The receiver picks a random number $b \in Z_p$ and calculates $B = g^b$ (if $r = 0$) or $B = Ag^b$ (if $r = 1$). He then sends B to the sender. The sender calculates $k_0 = H(B^a)$ and $k_1 = H((B/A)^a)$ such that k_0 and k_1 act as the secret keys in a symmetric encryption E. Then, he encrypts its messages, $e_0 = E_{k_0}(x_0)$, $e_1 = E_{k_1}(x_1)$, and sends the ciphertexts to the receiver. Then, the receiver calculates $k_R = H(A^b)$

Algorithm 1 OT extension Protocol Adopted from [13], $\mathcal{S}$ and $\mathcal{R}$ stand for sender and receiver

```
INPUT OF $\mathcal{S}$: $m$ tuples $(x_{j,0}, ..., x_{j,n-1})$ of $\ell$-bit messages, $1 \leq j \leq m$.

INPUT OF $\mathcal{R}$: $m$ selection integer $\mathbf{r} = (r_1, ..., r_m)$ such that $0 \leq r_j < n$ for $1 \leq j \leq m$.

COMMON INPUT:A security parameter $\kappa$ such that $\kappa \geq n$, and Walsh-Hadamard codes $\mathcal{C}^k_{\mathrm{WH}} = (\mathbf{c}_0, ..., \mathbf{c}_{k-1})$.

ORACLE: A random oracle H :$[m] \times \{0,1\}^\kappa \to \{0,1\}^\ell$.

CRYPTOGRAPHIC PRIMITIVE: A base $\mathrm{OT}^\kappa_m$ primitive.

  1. $\mathcal{S}$ chooses $\mathbf{s} \leftarrow \{0,1\}^\kappa$ at random. Let $s_i$ denote the $i^{th}$ bit of $\mathbf{s}$.
  2. $\mathcal{R}$ forms $m \times \kappa$ matrices $T_0, T_1$ in the following way:
       – Choose $\mathbf{t}_{j,0}$,$\mathbf{t}_{j,1} \leftarrow \{0,1\}^\kappa$ at random such that $\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1} = \mathbf{c}_{r_j}$
     Let $\mathbf{t}^i_0$,$\mathbf{t}^i_1$ denote the $i^{th}$ column of matrices $T_0, T_1$ respectively.
  3. $\mathcal{S}$ and $\mathcal{R}$ interact with $\mathrm{OT}^\kappa_m$ in the following way:
       – $\mathcal{S}$ acts as receiver with input $\{ s_i \}_{i \in [\kappa]}$.
       – $\mathcal{R}$ acts as sender with input $\{ \mathbf{t}^i_0, \mathbf{t}^i_1 \}_{i \in [\kappa]}$.
       – $\mathcal{S}$ receives output$\{ \mathbf{q}^i \}_{i \in [\kappa]}$.
     $\mathcal{S}$ forms m×k matrix Q such that the i-th column of Q is the vector $\mathbf{q}^i$.
     (Note $\mathbf{q}^i = \mathbf{t}^i_{s_i}$.)
     Let $\mathbf{q}_j$ denote the $j^{th}$ row of Q.
     (Note $\mathbf{q}_j = ((\mathbf{t}_{j,0} \oplus \mathbf{t}_{j,1}) \odot \mathbf{s}) \oplus \mathbf{t}_{j,0}$. Simplifying, $\mathbf{q}_j \oplus \mathbf{t}_{j,0} = \mathbf{c}_{r_j} \odot \mathbf{s}$.)
  4. For $j \in [m]$ and for every $0 \leq r < n$, $\mathcal{S}$ sends $y_{j,r} = x_{j,r} \oplus H$ ( j , $\mathbf{q}_j \oplus (\mathbf{c}_r \odot \mathbf{s})$).
  5. For $j \in [m]$, $\mathcal{R}$ recovers $z_j = y_{j,r} \oplus H(j, \mathbf{t}_{j,0})$.
```

and decrypts the desired message by its key as $x_r = D_{k_R}(e_r)$. H stands for a secure hash function.

The security of base OT protocols directly depends on the security of the underlying HE or DH cryptosystems. Since HE is a type of public key encryption, its security depends on the confidentiality of the private key, whereby only the owner of the private key has access to the content of the cipher-texts. DH key exchange protocol is secure due to the hardness of breaking Deffie-Hellman protocol and computing discrete logarithms.

Now, the results obtained from the execution of the base-OT are used to perform many OTs efficiently using lightweight symmetric operations.

The OT extension protocol proposed in [13] and shown in Algorithm 1 is the recent optimization of OT extensions that supports 1-out-of-n ($n > 2$) OT in addition to 1-out-of-2 OT. In step 3, base OTs is executed κ times. After performing base OTs, we can perform m number of 1-out-of-n OTs through steps 4 and 5. For proof of security, we refer to [13, Section 4]. In the following, we talk about the protocol's complexity since our approach is built on it.

The protocol executes OT_m^k, which has a complexity equal to that of OT_k^k (independent from m) plus generating $2k$ random strings which are each m bits long. In addition, each party evaluates at most mn times a random oracle. Therefore, the total communication of OT_l^m corresponds to communication complexity of OT_m^k plus mnl transferred bits between sender and receiver in Step 4, that is, $O(m(k + nl))$ bits. Consequently, the total computation complexity of the protocol is proportional to its communication complexity.

2.3. Arithmetic Sharing (AS)

Assuming distance functions consist of a series of addition and multiplication operations, we use Arithmetic sharing concept to calculate the distance privately.

The idea behind Arithmetic sharing is that a secret x is shared additively among two parties, P_0 and P_1, in the ring Z_{2^l} as x^0 and x^1 that satisfies $x^0 + x^1 = x \ mod \ Z_L$.

Given two shares x^i, y^i, it is possible to perform a privacy-preserving addition, subtraction and multiplication of the two corresponding secrets x, y. Due to the linear properties of Arithmetic sharing, the addition and subtraction of two secret-shared values can be computed locally as share-wise operations in the form of $x^0 + y^0 / \; x^0 - y^0$ and $x^1 + y^1 / \; x^1 - y^1$.

Similarly, a publicly known constant $c \in Z_{2^l}$ can be multiplied by the shares of a secret x, where $c \times x^0$ and $c \times x^1$ are computed locally.

In contrast, an interactive protocol is needed for multiplication of x and y ($x \times y$). The protocols for Arithmetic multiplication and our optimization are described in Section 4.1.

Assuming all Arithmetic operations are performed in mod 2^l, the concept of sharing and reconstruction is summarized as follows:

Sharing. P_0 chooses a random number $r \in Z_{2^l}$. It sets $x^0 = x - r$ and $x^1 = r$ then sends x^1 to P_1. This procedure is called $Shr(x)$ function.

Reconstruction. To reconstruct the secret value x, P_1 sends its share x^1 to P_0 who computes $x = x^0 + x^1$. This procedure is called $Rec(x)$ function.

3. Distance Measures

In the following sections, we introduce briefly the functionality of each distance measure.

3.1. Euclidean Distance (ED)

The squared Euclidean distance between two m-dimensional vectors $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_m\}$ is computed as $ED(X,Y) = \sum_{i=1}^{m} (x_i - y_i)^2$. Since the purpose of distance calculation for measuring the similarity between feature vectors is to find the closest match, and the exact value of the distance is not important, we avoid the square root of the distance in our privacy-preserving protocol.

Square root is a non-linear operation and cannot be addressed accurately through cryptographic techniques. Yu Bai et al. proposed an approximate calculation of square root; however, it introduces noise to the actual distance, in particular, when distance values are close to each other, the final result is not accurate[25].

3.2. *Cosine Similarity (CS)*

Cosine similarity measures the cosine of the angle between two feature vectors so the output value falls within $[0, 1]$.

The Cosine similarity between two m-dimensional vectors $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_m\}$ is computed as $CS(X, Y) = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{\mid X \mid\mid Y \mid}$ where $\mid X \mid$ and $\mid Y \mid$ are the norm of the vectors X and Y respectively ,i.e., $\mid X \mid = \sqrt{\sum_{i=1}^{m} x_i^2}$

3.3. *Scalar Product (SP)*

The scalar product between two m-dimensional vectors $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_m\}$ is computed as $SP(X, Y) = \sum_{i=1}^{m} x_i \cdot y_i$.

3.4. *Manhattan Distance (MD)*

The Manhattan distance between two m-dimensional vectors $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_m\}$ is computed as $MD(X, Y) = \sum_{i=1}^{m} |x_i - y_i|$.

3.5. *Mahalanobis Distance (MHD)*

The Mahalanobis distance between two m-dimensional vectors $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_m\}$ with covariance matrix S is computed as $MHD(X, Y) = \sqrt{(X - Y)^T S^{-1} (X - Y)}$.

3.6. *Edit Distance (EDD)*

Edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. In previous distance measures, the feature vectors are restricted to have equal length (m). However, in situations like measuring the similarity between strands of DNA, input vectors could have different lengths. Let's say the client owns m-length sequence X and the server has m'-length sequence Y, where each element of the sequences belongs to a finite alphabet set. A combination of insertions, deletions and substitutions can transform X into Y. The Edit distance is the minimum aggregate cost necessary to perform this transformation.

Intuitively, the complexity of the basic algorithm called Wagner-Fisher algorithm is $O(m \times m')$ or $O(m^2)$. This quadratic complexity makes the algorithm inefficient, particularly in the cryptography domain.

Ukkonen's algorithm improves upon the Wagner-Fisher by using a threshold k to limit the number of operations, provided that the Edit distance is less than a given threshold. It runs in $O(m \times k)$ time.

4. Our Proposed Protocols

In this section, we explain our proposed protocols to address secure Arithmetic multiplication and secure comparison.

4.1. Secure Arithmetic Multiplication

As shown in Sections 3.1 to 3.5, distance measures consist of a series of additions and multiplications. Addition can be done locally without need for interaction between the client and server. However, multiplication requires interactive protocols.

In the following sections, we describe three protocols including our proposed protocol for secure Arithmetic multiplication. Suppose P_0 holds x and P_1 holds y. They wish to execute a protocol by which P_0 obtains z^0 and P_1 obtains z^1 such that $z = z^0 + z^1$ and $z = x \cdot y$. We first describe our modification to the HE-based protocol proposed in [26]. Then we summarize the OT-based protocol proposed in [21]. Finally, we introduce our protocol which is a significant improvement over [21].

4.1.1. Arithmetic Multiplication using HE

Atallah et al. proposed a protocol in which values of x and y are additively secret-shared between P_0 and P_1 [26]. In our setting, however, P_0 holds the whole x and P_1 holds the whole y. Therefore, we use a slightly modified version of this protocol as shown in Algorithm 2 .

Algorithm 2 Arithmetic Multiplication using HE

```
$P_0 : x \in_R Z_{2^l}$
$P_1 : y \in_R Z_{2^l}, r \in_R Z_{2^{2l+\sigma}}, z^1 = -r$
$P_0 \rightarrow P_1 : [x]$
$P_1 \rightarrow P_0 : [d] = [x]^y.[r]$
$P_0 : z^0 = d$
```

Correctness Analysis: Since $z^0 = x \cdot y + r$ and $z^1 = -r$, $z^0 + z^1 = x \cdot y$.

Security Analysis: Because all messages received by P_1 are encrypted under public key of P_0, it cannot learn anything from $[x]$. P_0 cannot learn any information either, because it only receives blinded values in the form of $[d] = [x.y + r]$ that are statistically indistinguishable from uniformly random values selected from $Z_{2^{2l+\sigma}}$.

Communication Complexity: P_0 and P_1 exchange two ciphertexts ($[x]$ from P_0 to P_1 and $[d]$ from P_1 to P_0). Using the Paillier cryptosystem, each ciphertext has 2φ bits, so the total communication is 4φ bits. This protocol allows m number of Arithmetic multiplications using $2 \times m$ ciphertexts.

4.1.2. Arithmetic Multiplication using OT

Instead of using HE, Arithmetic multiplication can be performed based on the OT technique [21] as shown in Algorithm 3. Using OT is significantly faster than HE because symmetric operations, as opposed to public-key operations, are used in OT extension.

For an l-bit x, the 1-out-of-2 OT protocol should be executed l times. In the i^{th} execution P_0 receives $t_i^{a_i}$ from the pair (t_i^0, t_i^1). The correctness and security of this protocol have been proven in [21, Section 4.1].

Communication Complexity: The complexity of the secure multiplication protocol depends on the underlying OT protocol. Using the OT-extension pro-

Algorithm 3 Arithmetic Multiplication using OT

```
$P_0 : x \in_R Z_{2^l}$
    Binary representation of $x : x_{l-1}, ..., x_0$
$P_1 : b \in_R Z_{2^l}, s_0, ..., s_{l-1} \in_R Z_{2^l}$
    $t_i^0 = s_i, t_i^1 = 2^i \cdot y + s_i, , 0 \leq i \leq l-1$
    $z^1 = -\sum_{i=0}^{l-1} s_i$
$P_1 \rightarrow P_0 : 1 - out - of - 2 \text{ OT}(t_i^{x_i}), 0 \leq i \leq l-1$
$P_0 : c^0 = \sum_{i=0}^{l-1} t_i^{x_i}$
```

posed in [13], total communication is $l(\kappa + 2l)$ bits per multiplication.

4.1.3. Our Improved Protocol

While secure multiplication using OT is significantly faster than HE, we aim to further improve its efficiency. We developed a variant of OT based protocol which, instead of using a binary representation of x, is built on the hexadecimal representation of x. This new representation executes 1-out-of-16 OT four times less than Algorithm 3. So, if in the previous protocol 1-out-of-2 OT executes ρ times, in our improved protocol 1-out-of-16 OT executes $\rho/4$ times. From now on, $\rho/4$ is denoted by ρ'. As shown in Algorithm 4, our modification proceeds by following these steps:

1. P_1 selects ρ' random and independent elements denoted by $s_0, ..., s_{\rho'-1} \in_R Z_{2^\rho}$. It then prepares ρ' sets with sixteen elements in each: $(t_0^0, ..., t_0^{15}), ..., (t_{l'-1}^0, ..., t_{l'-1}^{15})$. For every $0 \leq i \leq l'-1$ and $0 \leq j \leq 15$, P_1 defines $t_i^j = j \cdot 16^i \cdot y + s_i$.
2. Considering the hexadecimal representation of x as $x_{l'-1}, ..., x_0$, P_0 and P_1 execute 1-out-of-16 OT l' times on l-bit messages. In the i^{th} execution, P_0 chooses $t_i^{a_i}$ from the set $(t_i^0, ..., t_i^{15})$.
3. P_0 sets $z^0 = \sum_{i=0}^{l'-1} t_i^{x_i}$ and P_1 sets $z^1 = -\sum_{i=0}^{l'-1} s_i$.

Correctness Analysis: Since $x_{l'}, ..., x_0$ is the hexadecimal representation of x, one can write $x = \sum_{i=0}^{l'-1} x_i \cdot 16^i$ as shown in Equation 1:

Algorithm 4 Improved Arithmetic Multiplication

```
$P_0 : x \in_R Z_{2^l}$
    Hexadecimal representation of $x : x_{l'-1}, ..., x_0$
$P_1 : y \in_R Z_{2^l}, s_0, ..., s_{l'-1} \in_R Z_{2^l}$
    $t_i^j = j \cdot 16^i \cdot y + s_i, 0 \leq i \leq l' - 1, 0 \leq j \leq 15$
    $z^1 = -\sum_{i=0}^{l'-1} s_i$
$P_1 \rightarrow P_0 : 1 - out - of - 16\ \mathrm{OT}(t_i^{x_i}), 0 \leq i \leq l' - 1$
$P_0 : z^0 = \sum_{i=0}^{l'-1} t_i^{x_i}$
```

$$\begin{aligned} z^0 + z^1 &= \sum_{i=0}^{l'-1} t_i^{x_i} - \sum_{i=0}^{l'-1} s_i \\ &= \sum_{i=0}^{l'-1} (x_i \cdot 16^i y + s_i) - \sum_{i=0}^{l'-1} s_i \\ &= y \cdot \sum_{i=0}^{l'-1} x_i \cdot 16^i \\ &= x \cdot y \end{aligned} \tag{1}$$

Security Analysis: This protocol ensures the privacy of each party given that they communicate via OT protocol. The only messages P_0 sends to P_1 are part of l' number of independent OTs. Also, the random and independent selection of $s_0, ..., s_{l'-1}$ ensures that the messages received by P_0 do not leak any information about other messages.

Communication Complexity: Using the OT-extension protocol proposed in [13], total communication takes $l'(\kappa + 16l)$ per Arithmetic multiplication.

Note that κ number of public-key operations are needed to perform base OTs for both Algorithms 3 and 4. Considering a 1024-bit Diffie-Hellman group, the communication for base OT takes $1024(\kappa + 1)$ bits.

4.1.4. Comparison of Arithmetic Multiplication Protocols

Table 1 compares the communication complexity of Arithmetic multiplication protocols described in Algorithm 2 to 4 with respect to different values of symmetric and public key security parameters (φ, κ) and different length of messages in bit (l) for one multiplication. As you can see from the $Improvement$ rows, our protocol outperforms two other approaches significantly, in particular for short messages like 8-bit messages. For example, in the presence of messages with 8-bit length our approach has about 90% improvement over Homomorphic based multiplication and about 50% improvement over 1-out-of-2 OT based multiplication. By increasing length of messages, we can see as we choose stronger security parameters ($\varphi = 2048$, $\kappa = 128$), our protocol outperforms its peers significantly.

4.2. Secure Comparison

Unlike distance measures composed of four basic mathematical operations $(+, -, \times, \div)$, Edit distance is based on Boolean comparison. It checks whether two specific characters from two separate sequences are equal or not. Therefore, we can reduce the problem of privacy-preserving Edit distance to secure comparison.

We propose a novel protocol for secure comparison based on OT. Let's say $X = \{x_0, ..., x_{m-1}\}$ and $Y = \{y_0, ..., y_{m'-1}\}$ are respectively the client and server's input sequences of characters in an alphabet set of size N. If we encode the characters as numbers, the code value vary from 0 to $N-1$. The goal of secure comparison is to check whether x_i and y_j are similar where $0 \leq x_i, y_j \leq N-1$.

In our proposed protocol, the client plays as the receiver and the server plays as the sender in OT. The OT messages are generated as follows:

$$M_{k(0\leq k\leq N-1)} = \begin{cases} 1 \quad if \qquad (k - y_j) \; mod \; N = 0 \\ 0 \quad else \end{cases} \tag{2}$$

On the other side, the receiver puts the value of x_i as its selection bit. Since the number of OT messages is n, execution of 1-out-of-n OT is required. The logic of this protocol is that if x_i and y_j are the same, then 1 will be transferred; otherwise, 0 will be transferred. Since the length of OT messages is one bit,

Table 1: Communication Complexity of Arithmetic multiplication protocols in terms of transferred bits between client and server

$\varphi = 1024$, $\kappa = 80$				
l	8	16	24	32
Algorithm 2 (Using HE)	4098	4098	4098	4098
Algorithm 3 (Using OT)	768	1792	3072	4608
Algorithm 4 (Our Approach)	416	1344	2784	4736
Improvement over Algorithm2	90%	67%	32%	–
Improvement over Algorithm3	46%	25%	10%	–
$\varphi = 1024$, $\kappa = 128$				
l	8	16	24	32
Algorithm 2 (Using HE)	4098	4098	4098	4098
Algorithm 3 (Using OT)	1152	2560	4224	6144
Algorithm 4 (Our Approach)	512	1536	3072	5120
Improvement over Algorithm2	87%	62%	25%	–
Improvement over Algorithm3	55%	40%	27%	16%
$\varphi = 2048$, $\kappa = 80$				
l	8	16	24	32
Algorithm 2 (Using HE)	8192	8192	8192	8192
Algorithm 3 (Using OT)	768	1792	3072	4608
Algorithm 4 (Our Approach)	416	1344	2784	4736
Improvement over Algorithm2	95%	83%	66%	42%
Improvement over Algorithm3	46%	25%	10%	–
$\varphi = 2048$, $\kappa = 128$				
l	8	16	24	32
Algorithm 2 (Using HE)	8192	8192	8192	8192
Algorithm 3 (Using OT)	1152	2560	4224	6144
Algorithm 4 (Our Approach)	512	1536	3072	5120
Improvement over Algorithm2	94%	81%	62%	37%
Improvement over Algorithm3	55%	40%	27%	16%

execution of the 1-out-of-n protocol proposed in [13] is highly recommended. It is the most efficient protocol known today for short-length messages, and we have adopted it in our implementation.

Correctness Analysis: The message corresponding to the selection bit is transferred (M_{x_i}). Intuitively, if $x_i = y_j$ then the condition $(k - y_j)\ mod\ N = 0$ is satisfied and the message is 1. If $x_i \neq y_j$ then the transferred message is 0.

Security Analysis: The security of secure comparison protocol depends on the security of the underlying OT protocol. The receiver only receives the message corresponding to its selection bit and gains no information about the other messages. The sender also does not learn anything about the selection bit. In the Edit distance algorithm, when several comparisons are required, only sequences with enough similarity (based on the threshold k) are processed to the end. If the Edit distance exceeds the threshold then the execution will stop. This way, we can minimize the information leakage. Empirically, we set the value of k to be 60, 80 and 100 in our experiments. Since the goal of privacy preserving Edit distance in genome related scenarios is to return the top most similar patterns, through exploration of the dataset provided by [27] we found that these values for the threshold cause the algorithm to return reasonable number of similar patterns (about 10). However, in our implementation the value of the threshold is an input parameter that can be varied based on the scenario. Also, the average length of genome patterns is 3,500 characters so that a threshold between 60 to 100 will return the most similar patterns i.e., patterns which only need 60 to 100 transformations to replicate each other. On the other hand, for short patterns the threshold should be a smaller value for a good trade-off between accuracy and information leakage.

Communication Analysis: For each comparison, the communication bandwidth takes $\kappa + N$ bits where N is database size. Edit distance algorithm requires $k \times m$ number of comparisons so the consumption of the bandwidth is $(k \times m) \times (\kappa + N)$ bits. Concretely, the value of threshold k impacts communication complexity linearly. In our experiments, we evaluate the privacy preserving Edit distance algorithms with different value of k as 60, 80 and 100.

5. privacy-preserving Distance Measures

We have described the building blocks of our framework so far. In this section, we show how privacy-preserving distance measures work and in particular report the experimental results for Euclidean distance, Cosine Similarity and Edit distance. It should be noted that our framework is a **Java-based** implementation leading to longer running time compared to C++ implementation of cryptographic protocols. Hence, we focus on the communication complexity and number of required computational operations not absolute execution time.

5.1. Experimental Setup

We evaluate our approach with respect to execution time and communication bandwidth. The goal of performance evaluation is to show the feasibility of our approach in real-world identification systems. The general framework has been implemented in Java and the client and server communicate through sockets. We run the framework over a WAN network rather than a LAN network to have a better approximation of real-world scenarios. All the experiments are the average of 10 execution rounds. To do this, we use an intercontinental cloud setting and perform the experiments on two free-tier Amazon instances with a 64-bit Intel Xeon dualcore CPU with 2.8 GHz and 3.75 GB RAM. The client and server are located in Oregon and Tokyo respectively. The source code is available [2] .

5.1.1. database

The client's input and server's database are generated randomly and feature values are 8-bits numbers in the range $[0, 255]$. For Edit distance, we evaluate our protocol using a genome database released by "iDASH Security and Privacy Workshop" [27].

[2] *http : //people.tamu.edu/ kaghazgaran/Privacy_Preserving_Distance_Calculation_Framework.7z*

5.1.2. Security Parameters

We evaluate our system with different public key security parameters $\varphi \in \{1024, 2048\}$. The symmetric security parameter κ is set to be 80 or 128. Although the related work that are based on 1-out-of-2 OT set security parameters as $\varphi = 1024$ and $\kappa = 80$ [17] , we evaluate feasibility of our protocol in terms of both computation and communication in presence of stronger security parameters by considering key security parameters with longer bit-length that is $\varphi = 2048$ and $\kappa = 128$.

5.2. privacy-preserving Euclidean Distance

Euclidean distance is the most widely used distance measure in privacy-preserving biometric identification approaches.

Our proposed protocol for ED is based on additive and multiplicative Arithmetic sharing. The computation of *ED* can be broken into three parts:

$$ED(X,Y) = \overbrace{\sum_{i=1}^{m} x_i^2}^{1} - 2 \times \overbrace{\sum_{i=1}^{m} (x_i . y_i)}^{2} + \overbrace{\sum_{i=1}^{m} y_i^2}^{3} \tag{3}$$

Parts 1 and 3 are calculated locally by the client and server respectively. Then the client and server jointly run secure Arithmetic multiplication as shown in Algorithm 4 m times where m is feature vector size. The client obtains one additive share of the distance as $D^0 = \sum_{i=1}^{m}(x_i^2 - 2z_i^0)$ and the sever obtains the other share of the distance as $D^1 = \sum_{i=1}^{m}(y_i^2 - 2z_i^1)$.

This protocol can be easily extended to 1-to-many biometric identification by executing it N times where N indicates the size of the database. If we deal with integer values for x_i and y_i then privacy-preserving approach will not affect the accuracy of the results and the private distance is equal to actual value of the distance.

5.2.1. Performance Evaluation of Privacy Preserving Euclidean Distance

We evaluate the Euclidean distance protocol with different bit lengths of the OT messages (l) and database sizes.

Figures 2 and 3 show the execution time and bandwidth respectively when $\varphi = 1024$ and $\kappa = 80$ when length of OT messages varies from 8-bit to 32-bit in the presence of different database size that is 128, 256, 320 and 512 patterns. It should be noted that each line in the diagrams corresponds to a specific database size. For example, when the size of the database is 256 and the bit length of the OT messages is 32 bits, the privacy-preserving identification protocol takes only 3.6 seconds on the WAN network. This time also includes communication time that means if we execute our protocol on a LAN network it would take less time. The communication bandwidth varies from 190 KB to 1400 KB depending on the message bit length and database size. As we can see in Figures 2 and 3, the execution time does not scale proportionally when we double up the size of the database. For example, the execution time is 3 sec and 3.2 sec in the presence of databases with 128 and 320 in size respectively when message length is 8-bit.

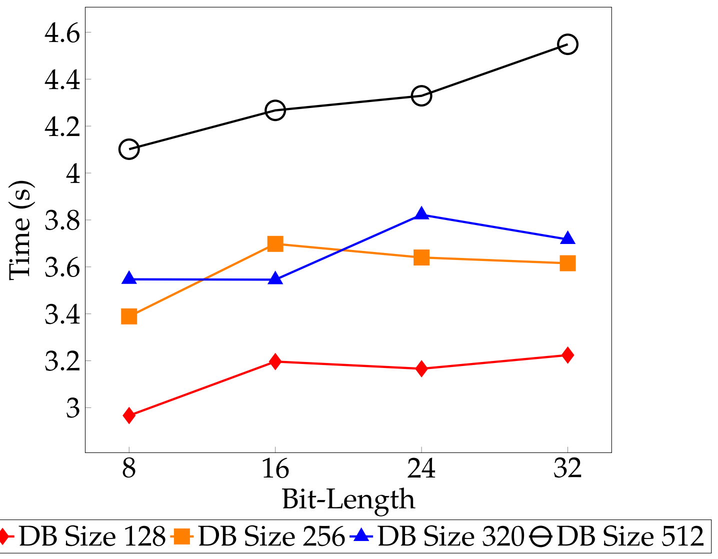

Figure 2: Privacy-Preserving Euclidean Distance Execution Time when $\varphi = 1024$ and $\kappa = 80$

Figures 12 and 5 shows the execution time and bandwidth respectively when $\varphi = 1024$ and $\kappa = 128$. For example, when the size of the database is 256 and the bit length of the OT messages is 32 bits, the privacy-preserving

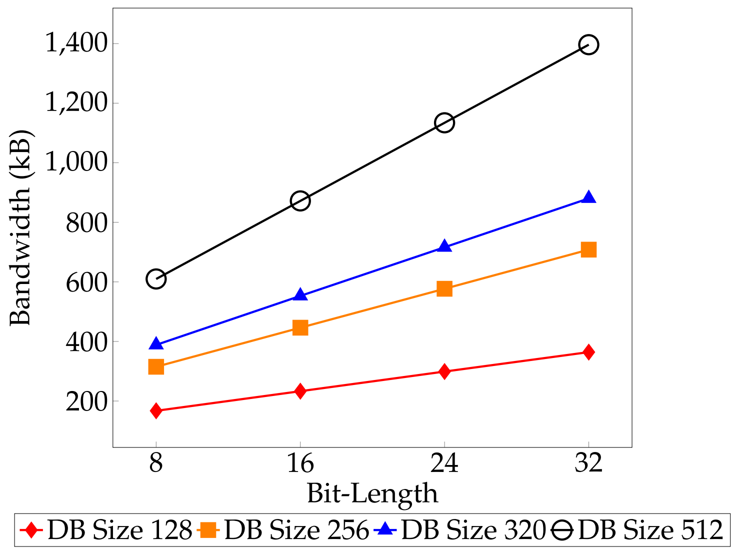

Figure 3: Privacy-Preserving Euclidean Distance Bandwidth when $\varphi = 1024$, $\kappa = 80$

identification protocol takes only 4.2 seconds on the WAN network. The communication bandwidth varies from 200 KB to 1600 KB depending on the message bit length and database size.

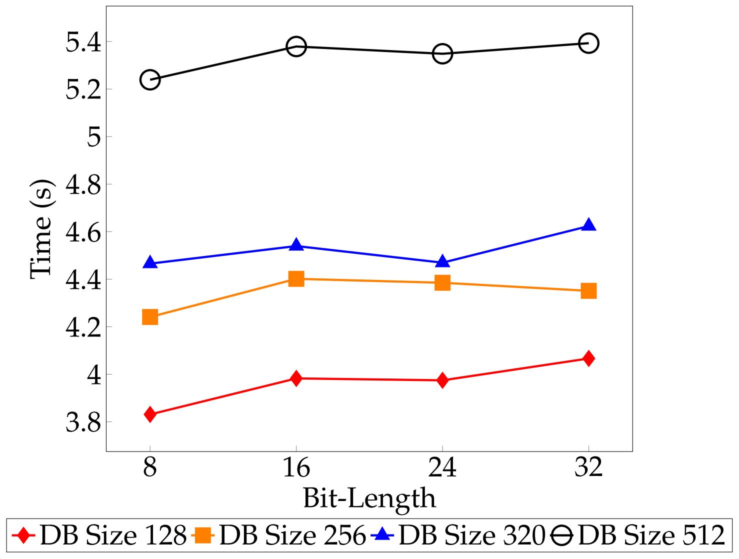

Figure 4: Privacy-Preserving Euclidean Distance Execution Time when $\varphi = 1024$, $\kappa = 128$

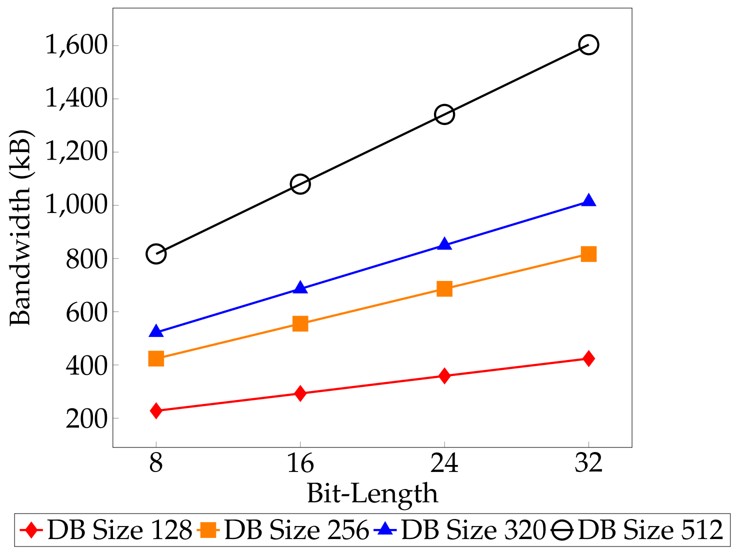

Figure 5: Privacy-Preserving Euclidean Distance Bandwidth when $\varphi = 1024$, $\kappa = 128$

Table 2: Performance Evaluation of Privacy-preserving Euclidean Distance when $\varphi = 2048$

	Execution Time (s)				Communication(KB)			
l	8	16	24	32	8	16	24	32
$\kappa = 80$	10	9.1	8.9	10	182	248	313	379
$\kappa = 128$	12.2	13.6	12.5	12.2	252	318	383	449

We also execute our protocol with $\varphi = 2048$ and keep the database size at 128. Table 2 shows the results in terms of execution time and bandwidth.

5.3. *Privacy-preserving Cosine Similarity*

Since cosine similarity includes the division operation and cryptographic techniques only support addition and multiplication operations, securely computing cosine similarity is not straightforward. For simplification, taking the normalization of X and Y, one can reduce the computational cost of cosine similarity as CS($X/\mid X\mid, Y/\mid Y\mid) = X' \cdot Y'$ where $\mid X\mid$ and $\mid Y\mid$ are $\sqrt{\sum_{i=0}^{m-1} x_i^2}$ and $\sqrt{\sum_{i=0}^{m-1} y_i^2}$ respectively.

The client and server first divide each feature value by their vector normalization value to get rid of division so the calculation of cosine similarity will be reduced to the calculation of $CS(X, Y) = \sum_{i=1}^{m} x_i' \cdot y_i'$ where x_i' and y_i' are

$x_i / \mid X \mid$ and $y_i / \mid Y \mid$ respectively. On the other hand, these divisions give us real numbers while cryptographic operations only deal with integer numbers. To address this issue, we multiply all the feature values by a factor of 10 such as 10, 100, 1000 or 10000 then round the results to the nearest integer values. We call this process *Scaling* and empirically show that 1000 is the best scaling factor.

After reducing Cosine similarity to dot product, the client and server jointly execute Algorithm 4 m times. At the end of the protocol, the client obtains one additive share of the distance $D^0 = \sum_{i=1}^{m} Z_i^0$ and the server obtains other share of the distance $D^1 = \sum_{i=1}^{m} Z_i^1$.

Accuracy Analysis: As we mentioned earlier, to get rid of the real numbers and preserve the accuracy of the final result to some extent, the normalized feature values first are multiplied by a factor of 10 and then are rounded to their nearest integer value (*Scaling* process). It should be noted that transforming non-integers into integers through simple rounding would be inaccurate and thus ineffective. For example, in our case, the normalized feature values are bounded in the range [0, 1], the transformed values will all be either zero or one, which makes the distance calculation quite useless.

To figure out the best scaling factor, a balance between accuracy and complexity should be achieved. For this purpose, we run our protocol 50 times per scaling factor and calculate the average error rate, execution time and bandwidth for 32 bit messages, database with 128 size, $\kappa = 80$ and $\varphi = 1024$ as shown in Table 3. We can see that when scaling factor is 1000 we can achieve the best trade-off between accuracy and performance i.e., reasonable error rate and communication overhead. Equation 4 shows how the error rate is calculated. The error rate evaluates the impact of *Scaling* process on accuracy.

$$ErrorRate = \frac{\mid Actual \quad Distance - Private \quad Distance \mid}{Actual \quad Distance} \times 100 \qquad (4)$$

Table 3: Cosine Similarity Accuracy Analysis

Factor	Execution Time (s)	Communication(KB)	Error Rate
10	2.9	192	35.4
100	3.2	364	4.8
1000	3.4	536	0.45
10000	3.8	708	0.08

5.3.1. Performance Evaluation of Privacy Preserving Cosine Similarity

We evaluate the Cosine similarity protocol with OT messages of different bit-lengths (l) and different database sizes using 1000 as a scaling factor to achieve an acceptable accuracy-efficiency trade-off.

Figures 6 and 7 show the execution time and bandwidth respectively when $\varphi = 1024$ and $\kappa = 80$. For example, when the size of the database is 256 and the bit-length of the OT messages is 32 bits, the privacy-preserving identification protocol takes only 5.2 seconds on the WAN network. The communication bandwidth varies from 200 KB to 2200 KB depending on the message bit-length and database size.

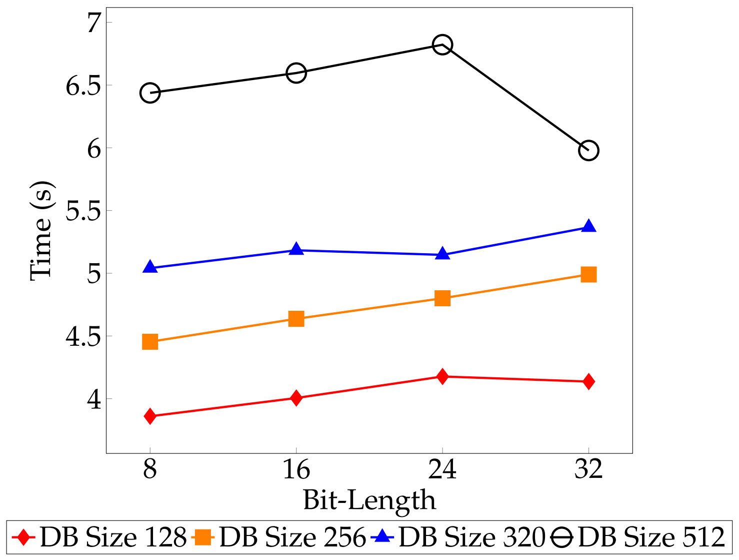

Figure 6: Privacy-Preserving Cosine Similarity Execution Time when $\varphi = 1024$, $\kappa = 80$

Figures 8 and 9 show the execution time and bandwidth respectively when

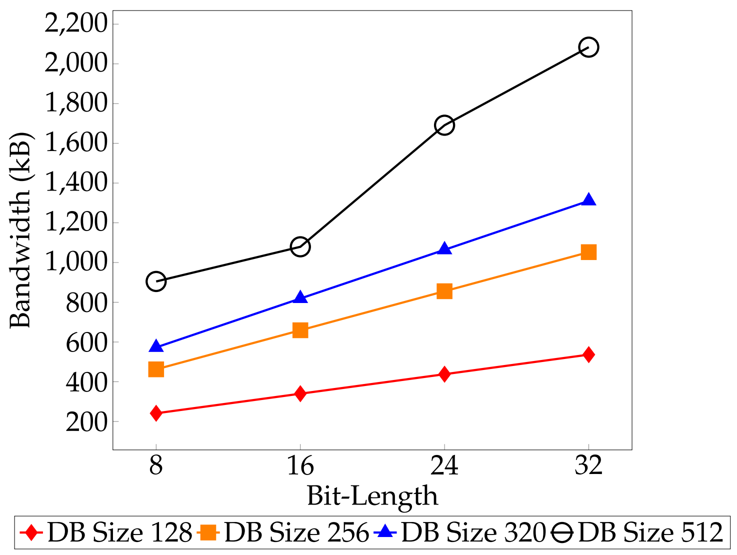

Figure 7: Privacy-Preserving Cosine Similarity Bandwidth when $\varphi = 1024$, $\kappa = 80$

Table 4: Performance Evaluation of Privacy-Preserving Cosine Similarity when $\varphi = 2048$

	Execution Time (s)				Communication(KB)			
l	8	16	24	32	8	16	24	32
$\kappa = 80$	11	11	10.3	9.2	257	355	454	552
$\kappa = 128$	12.6	13.5	13.4	14	350	449	547	645

$\varphi = 1024$ and $\kappa = 128$. For example, when the size of the database is 256 and the bit-length of the OT messages is 32 bits, the privacy-preserving identification protocol takes only 6.7 seconds on the WAN network. The communication bandwidth varies from 300 KB to 2400 KB depending on the message bit length and database size.

We also execute our protocol with $\varphi = 2048$ and keep the database size at 128. Table 4 shows the results in terms of execution time and bandwidth.

5.4. Privacy-preserving Edit Distance

To calculate Edit distance between two sequences, our proposed secure comparison protocol executes $k \times m$ times, where k is the distance threshold and m is the maximum length of the input sequences.

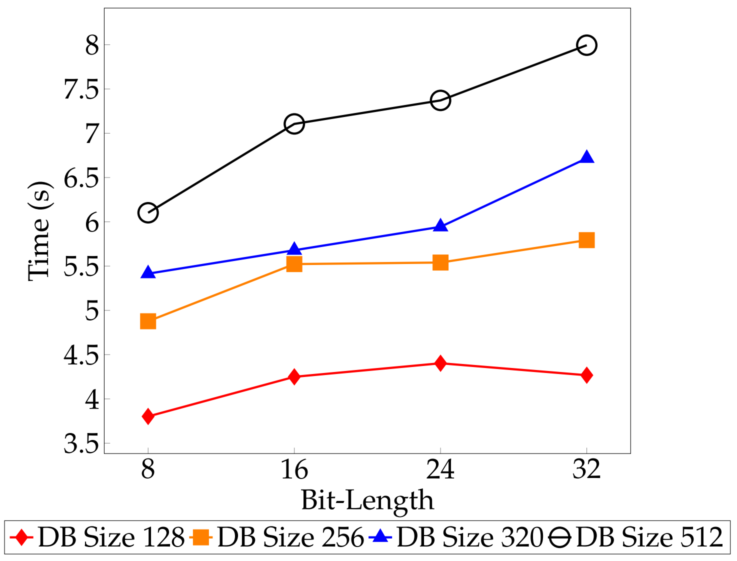

Figure 8: Privacy-Preserving Cosine Similarity Execution Time when $\varphi = 1024$, $\kappa = 128$

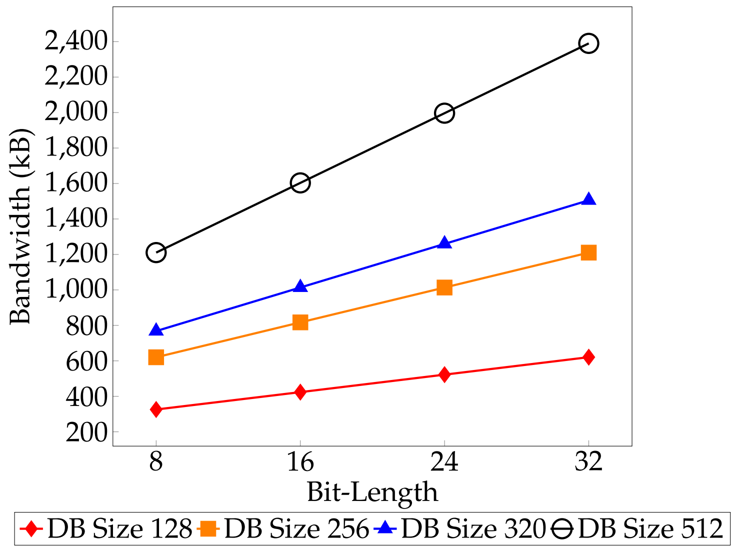

Figure 9: Privacy-Preserving Cosine Similarity Bandwidth when $\varphi = 1024$, $\kappa = 128$

5.4.1. *Performance Evaluation of Privacy Preserving Edit Distance*

490 To evaluate our approach, we use the database of genome data [27] in which the server holds 50 different sequences. The length of the sequences in average is 3500 characters from $\{A, C, G, T\}$ alphabet set. The experimental results are

shown in Figures 10 and 11 with different security parameters.

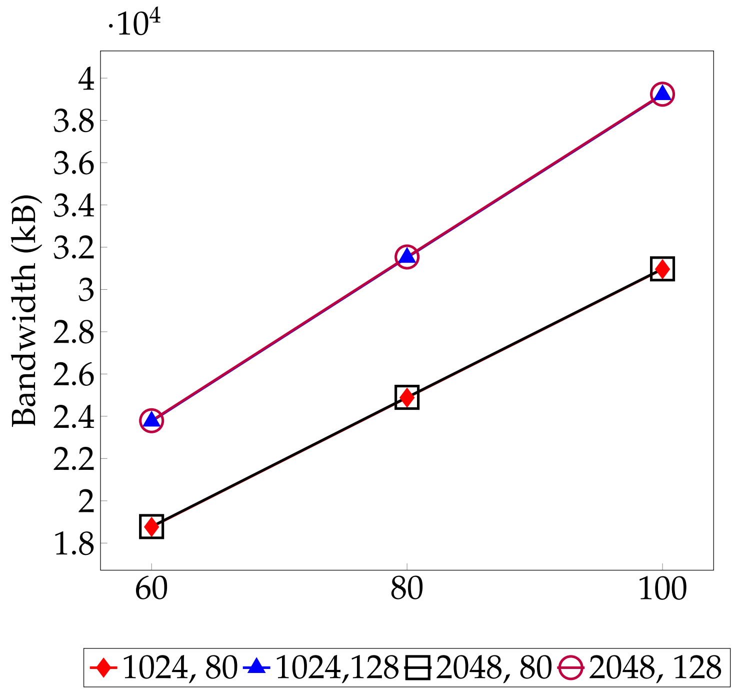

Figure 10: Privacy-Preserving Edit Distance Bandwidth

We also run the our Edit distance framework over LAN network in addition to WAN network. For LAN setting, we use the VM machines provided by "iDASH Security and Privacy Workshop 2016" so, we can provide a fair comparison with state of the art work [28] which run the experiments on the same VM machines. Figure 12 demonstrates the execution time over LAN nework.

Results: We set Edit distance threshold k to 60, 80 and 100. The goal is to return the sequences with equal or less than the threshold k dissimilarity to the client sequence. Obviously, by increasing the threshold the complexity $O(k \times m)$ increases. Experimental results are shown in Figures 10 to 12 with different security parameters ($\varphi \in \{1024, 2048\}$, $\kappa \in 80, 128$). Figures 11 and 12 measures the running time in second on LAN and WAN network respectively while Figure 10 shows the bandwidth consumption in KB. Execution time varies from 8 to 38 seconds on LAN and 45 to 75 seconds on WAN. The

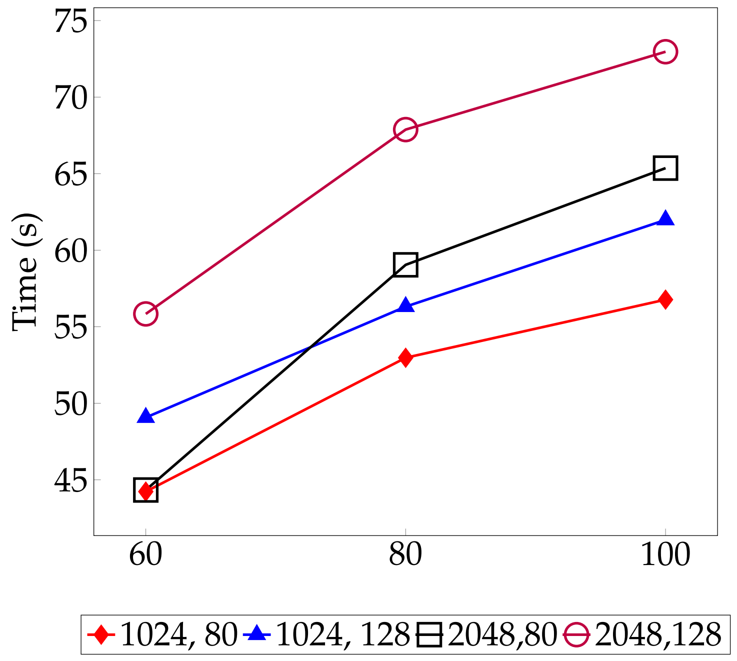

Figure 11: Privacy-Preserving Edit Distance Execution Time over WAN network

execution time on WAN is higher due to network latency. Bandwidth directly depends on symmetric security parameter κ or number of base-OTs, as it is shown in Figure 10 the public-key security parameter φ does not affect the communication. The bandwidth varies from 18 to 40 MB.

Analysis: The private Edit distance protocols proposed in [28] is executed in 23 seconds on the same dataset and over the same virtual machines with baseline security parameters ($\varphi = 1024$ and $\kappa = 80$). While our proposed protocol runs only in 8 seconds with same configuration. The other advantage of our approach over [28] is that ***ESCOT*** protocol calculates accurate Edit distance while the other work approximates the Edit distance.

6. privacy-preserving Comparison

In Euclidean distance and Cosine similarity protocols, the client and server obtain an additive share of the actual distance at the end of the protocols. A

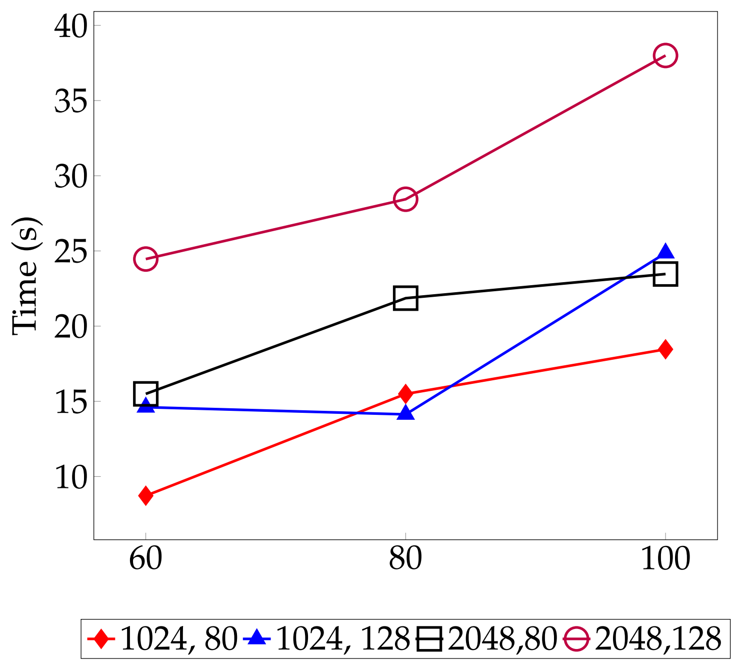

Figure 12: Privacy-Preserving Edit Distance Execution Time over LAN network

simple solution is that the server sends its share to the client and the client reconstructs the actual distance. However, revealing the actual value of distance to one of the parties makes the privacy-preserving protocol worthless. The fundamental assumption of secure computation protocols says no information but the final result should be revealed.

Therefore, the next step is to run a comparison protocol to obtain desired results. There are three approaches to privacy-preserving comparison:

1. The identity of the record with minimum distance is returned to the client [5, 6, 7, 17, 29].
2. All the identities with distance values less than a specific threshold are returned to the client [20, 3, 8].
3. The client learns the comparison results between its input and all the records in the server's database [4].

There are some works that use different techniques for comparison such as

Homomorphic encryption [5, 3], Garbled circuit [6, 7, 4, 8, 29] and GMW [17].

While in the first two distance measures we need a comparison protocol, Edit distance has a different story and the client obtains identities of the records whose distance to the client's input is less than a given threshold.

7. Related Work

In this section, we describe related work with regard to each of the distance measure.

Euclidean Distance: The most portion of related work are built on Homomorphic encryption. Erkin et al. proposed the first privacy-preserving Euclidean distance protocol for face recognition using the Paillier cryptosystem [5]. This protocol calculates Euclidean distances between the client's encrypted input and all records in the server's database. Sadeghi et al. improved the previous protocol by devising the idea of packing to optimize communication bandwidth [6]. Barni et al. proposed a privacy-preserving protocol for fingerprint identication using the Paillier Homomorphic encryption [3]. Evans et al. proposed a privacy-preserving biometric identication protocol with a focus on fingerprint data [7]. This protocol optimizes the complexity of the Paillier cryptosystem by proposing a new technique for packing. Blanton et al. [4] used DGK homomorphic cryptosystem for fingerprint identification. Chun et al. proposed a protocol in which data and computations are outsourced to a cloud [8]. The server's database and the client's input are encrypted rst and then sent to a cloud. This approach also uses the Paillier cryptosystem. Since all the computations are done on encrypted domain, the experimental results show it is not practical. All these approaches are restricted by the inefficiency of underlying homomorphic encryption.

Cosine Similarity: To our knowledge, there exist two works addressing privacy-preserving cosine similarity using homographic encryption [30, 31]. The proposed approach in [30] is theoretical and does not address the issue of real numbers in practice. The protocol is built upon ElGamal homomorphic encryption and zero knowledge proof. The final result just shows the cosine similarity as 1 or 0 which is not accurate and causes huge information loss. The proposed approach in [31] is based on HE and only guarantees the privacy of the server's database. The server's database is stored in encrypted form and client input is sent in plain to the server.

Edit Distance : Shantanu and Boufounos proposed an approach to calculate Edit distance using HE [32]. They reduced the problem to privacy-preserving minimum finding protocol that should be executed $m \times m'$ times (m and m' are the length of the input sequences). Huang et al. proposed a protocol to calculate Edit distance based on Garbled circuits [10].

8. Conclusion and Future Work

In this paper, we develop a general framework to address privacy-preserving distance calculation in a efficient way. Our main method is Oblivious Transfer. The security of our protocol is directly based on the security of the Oblivious Transfer scheme. We evaluate our approach in terms of execution time and communication bandwidth. More importantly, we address the privacy-preserving Edit distance using OT for the first time. Now that the client and server have an additive share of the distances, the final goal is to find the minimum distance or distances less than a specic threshold. In our future work, we will consider privacy-preserving comparison protocol. Also, we will consider the malicious adversary model by proposing our protocols based on committed Oblivious Transfer.

References

[1] A. C.-C. Yao, How to generate and exchange secrets, in: Foundations of Computer Science, 1986., 27th Annual Symposium on, IEEE, 1986, pp. 162–167.

[2] S. Goldwasser, S. Micali, A. Wigderson, How to play any mental game, or

a completeness theorem for protocols with an honest majority, in: Proc. of the Nienteenth Annual ACM STOC, Vol. 87, 1987, pp. 218–229.

[3] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, et al., Privacy-preserving fingercode authentication, in: Proceedings of the 12th ACM workshop on Multimedia and security, ACM, 2010, pp. 231–240.

[4] M. Blanton, P. Gasti, Secure and efficient protocols for iris and fingerprint identification, in: Computer Security–ESORICS 2011, Springer, 2011, pp. 190–209.

[5] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, T. Toft, Privacy-preserving face recognition, in: Privacy Enhancing Technologies, Springer, 2009, pp. 235–253.

[6] A.-R. Sadeghi, T. Schneider, I. Wehrenberg, Efficient privacy-preserving face recognition, in: Information, Security and Cryptology–ICISC 2009, Springer, 2009, pp. 229–244.

[7] D. Evans, Y. Huang, J. Katz, L. Malka, Efficient privacy-preserving biometric identification, in: Proceedings of the 17th conference Network and Distributed System Security Symposium, NDSS, 2011.

[8] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, W. Jiang, Outsourceable two-party privacy-preserving biometric authentication, in: Proceedings of the 9th ACM symposium on Information, computer and communications security, ACM, 2014, pp. 401–412.

[9] J. Bringer, M. Favre, H. Chabanne, A. Patey, Faster secure computation for biometric identification using filtering, in: 2012 5th IAPR International Conference on Biometrics (ICB), IEEE, 2012, pp. 257–264.

[10] Y. Huang, D. Evans, J. Katz, L. Malka, Faster secure two-party computation using garbled circuits., in: USENIX Security Symposium, Vol. 201, 2011.

[11] Y. Luo, S.-c. S. Cheung, T. Pignata, R. Lazzeretti, M. Barni, An efficient protocol for private iris-code matching by means of garbled circuits, in: 2012 19th IEEE International Conference on Image Processing, IEEE, 2012, pp. 2653–2656.

[12] G. Asharov, Y. Lindell, T. Schneider, M. Zohner, More efficient oblivious transfer and extensions for faster secure computation, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM, 2013, pp. 535–548.

[13] V. Kolesnikov, R. Kumaresan, Improved ot extension for transferring short secrets, in: Advances in Cryptology–CRYPTO 2013, Springer, 2013, pp. 54–70.

[14] D. Demmler, T. Schneider, M. Zohner, Aby-a framework for efficient mixed-protocol secure two-party computation., in: NDSS, 2015.

[15] T. Schneider, Practical aspects of secure two-party computation, in: Summer School on Secure and Trustworthy Computing, TU Darmstadt, 2015, pp. 1–65.

[16] J. Bringer, H. Chabanne, A. Patey, Shade: Secure hamming distance computation from oblivious transfer, in: Financial Cryptography and Data Security, Springer, 2013, pp. 164–176.

[17] J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, M. Zohner, Gshade: Faster privacy-preserving distance computation and biometric identification, in: Proceedings of the 2nd ACM workshop on Information hiding and multimedia security, ACM, 2014, pp. 187–198.

[18] T. Schneider, Aby - a framework for efficient mixed-protocol secure two-party computation, in: Securing Computation Workshop, EC SPRIDE, 2015, pp. 1–18.

[19] A. Jarrous, B. Pinkas, Secure hamming distance based computation and its applications, in: Applied Cryptography and Network Security, Springer, 2009, pp. 107–124.

[20] M. Osadchy, B. Pinkas, A. Jarrous, B. Moskovich, Scifi-a system for secure face identification, in: Security and Privacy (SP), 2010 IEEE Symposium on, IEEE, 2010, pp. 239–254.

[21] N. Gilboa, Two party rsa key generation, in: Annual International Cryptology Conference, Springer, 1999, pp. 116–129.

[22] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, Recommendation for key management-part 1: General (revision 3), 2012, NIST Special Publication 800–57.

[23] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1999, pp. 223–238.

[24] T. Chou, C. Orlandi, The simplest protocol for oblivious transfer, in: International Conference on Cryptology and Information Security in Latin America, Springer, 2015, pp. 40–58.

[25] Y. Bai, L. Zhuo, B. Cheng, Y. F. Peng, Surf feature extraction in encrypted domain, in: Multimedia and Expo (ICME), 2014 IEEE International Conference on, IEEE, 2014, pp. 1–6.

[26] M. Atallah, M. Bykova, J. Li, K. Frikken, M. Topkara, Private collaborative forecasting and benchmarking, in: Proceedings of the 2004 ACM workshop on Privacy in the electronic society, ACM, 2004, pp. 103–114.

[27] in: AMIA ANNUAL FALL SYMPOSIUM, GENOPRI WORKSHOP, http://www.humangenomeprivacy.org/2016/competition-tasks.html, 2016.

[28] M. M. Al Aziz, et al., Secure approximation of edit distance on genomic data, BMC Medical Genomics.

[29] R. Lazzeretti, M. Barni, Private computing with garbled circuits [applications corner], Signal Processing Magazine, IEEE 30 (2) (2013) 123–127.

[30] D. Yang, C. Lin, B. Yang, A novel secure cosine similarity computation scheme with malicious adversaries, International Journal of Network Security & Its Applications 5 (2) (2013) 171.

[31] H. Kikuchi, K. Nagai, W. Ogata, M. Nishigaki, Privacy-preserving similarity evaluation and application to remote biometrics authentication, Soft Computing 14 (5) (2010) 529–536.

[32] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, R. Sirdey, Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain, Signal Processing Magazine, IEEE 30 (2) (2013) 108–117.

Parisa Kaghazgaran is Ph.D. student in Computer Science and Engineering at Texas A&M University. She has done a broad research on security topics such as Cryptography protocols, Privacy, and Insider Threat. She has been doing research on intersection of social media problems such as misbehavior and fake reviews and machine learning algorithms since 2016. She has been a research intern at Microsoft Research (MSR) in summer 2018. She is student member of IEEE and ACM. Contact her at parisakaghazgaran@my.unt.edu.

Hassan Takabi is an Assistant Professor of Computer Science and Engineering at the University of North Texas, Denton, Texas, USA. He is director and founder of the INformation Security and Privacy: Interdisciplinary Research and Education (INSPIRE) Lab and a member of the Center for Information and Computer Security (CICS), which is designated as National Center for Academic Excellence in Information Assurance Research (CAE-R) and Education (CAE-IAE). His research is focused on various aspects of cyber-security and privacy including advanced access control models, insider threats, cloud computing security, mobile privacy, privacy and security of online social networks, and usable security and privacy. He is member of ACM and IEEE. Contact him at takabi@unt.edu.

Flannery Currin is an undergraduate student at Earlham College majoring in Computer Science and Psychology. During the summer of 2017, she worked at the Pacific Northwest National Laboratory as part of the National Security Internship Program and then returned to Earlham to assist a professor with research and course design. She spent the summer of 2016 working at the University of North Texas as part of a Research Experience for Undergraduates program focused on software testing and security.

Armando Soriano is a B.S. in Computer Science student at Tarleton State University. He is working and attending university as a full-time student as well as being a research assistant for his Computer Science department. He participated in the REU: Bug Wars program during the summer of 2016 at the University of North Texas under the supervision of Parisa Kaghazgaran working on privacy enhancing techniques.