Communication-efficient private distance calculation based on oblivious transfer extensions

Parisa Kaghaazgaran\textsuperscript{a,}\textsuperscript{*}, Hassan Takabi\textsuperscript{b}, Flannery Hope Currin\textsuperscript{b}, Armando Soriano\textsuperscript{b}

\textsuperscript{a}Texas A&M University, College Station, TX, USA
\textsuperscript{b}University of North Texas, Denton, TX, USA

\textbf{A R T I C L E   I N F O}

Article history:
Received 20 June 2018
Revised 24 December 2018
Accepted 25 January 2019
Available online 30 January 2019

Keywords:
Private identification
Oblivious transfer
Distance-based similarity
Efficiency
Communication complexity

\textbf{A B S T R A C T}

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 the client and the 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. We use the semi-honest adversary threat model. We extensively evaluate our PPDM framework. And, we theoretically show the improvement of PPDM over related work.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Secure Two-party Computation, introduced by Yao (1986) and Goldreich-Micali-Wigderson (GMW) (Goldwasser et al., 1987), solves problems in which two parties want to jointly evaluate a function over 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 (Barni et al., 2010; Blanton and Gasti, 2011; Chun et al., 2014; Erkin et al., 2009; Evans et al., 2011; Sadeghi et al., 2009). 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 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 like the one proposed by Erkin et al. (2009). 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 (Barni et al., 2010; Blanton and Gasti, 2011; Evans et al., 2011; Sadeghi et al., 2009) or pure circuit-based techniques (Bringer et al., 2012; Huang et al., 2011; Luo et al., 2012).

While it has been shown (Asharov et al., 2013; Demmler et al., 2015; Kolesnikov and Kumaresan, 2013) that Oblivious Transfer (OT) is the fastest secure technique in a two
party setting, particularly in privacy-preserving biometric identification (Bringer et al., 2014; 2013; Schneider, 2015b), communication bandwidth and multiple communication rounds are the main bottlenecks 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 (Schneider, 2015a).

A few recent works have improved the complexity of OT-based protocols (Asharov et al., 2013; Kolesnikov and Kumaresan, 2013) and evaluated their applications in the privacy-preserving biometric identification problem (Demmler et al., 2015). In this paper, we build upon existing protocols to improve communication complexity in OT-based protocols by using 1-out-of-n OTs. Note 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. (2013) proposed a protocol for privacy-preserving Hamming distance using OT. Then, Bringer et al. (2014) generalized this protocol to enable privacy-preserving computation of other distance metrics using OT. Both works 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 contribution 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.

It should be noted that in this work we deal with two different types of distance functions. One includes functions like Euclidean distance and cosine similarity that consist of a series of addition and multiplication operations. To address distance functions of this type, we propose to calculate Arithmetic operations in the privacy-preserving setting (Section 4.1). The second type of the distance functions, including Edit distance, aim to quantify the similarity between two strings of characters (discrete values) and are based on comparison between those discrete values (they are either equal or different). To address distance functions of this type we propose privacy-preserving comparison protocols (Section 4.2).

We skip Hamming distance since it deals exclusively with binary inputs and has been addressed many times using different techniques such as Homomorphic encryption (Jarrous and Pinkas, 2009; Osadchy et al., 2010), Garbled circuits (Huang et al., 2011) and Oblivious Transfer (Bringer et al., 2013).

Formal definition: As shown in Fig. 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 Y1, . . . ,YN. Then, they jointly calculate the distance between X and each Yi. At the end of the protocol, each party has an additive share of the distance. Suppose D indicates the actual distance, the client obtains D0 and the server obtains D1 such that D0 + D1 = D. The distance shares are then fed to a privacy-preserving comparison protocol to find the best match. The focus of this paper is to propose an efficient protocol to calculate the distance shares in a private manner.

1.1. Our contributions

We break down 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 rather than 1-out-of-2 OT (Demmler et al., 2015; Gilboa, 1999) 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 online1.

2. Background information

In this section, we first introduce the notations used throughout the paper. We then 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 of the previous work.

---

1 http://people.tamu.edu/~kaghazgaran/Privacy_Preserving_Distance_Calculation_Framework7z.
2.1. 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

\begin{algorithm}
\caption{OT extension Protocol Adopted from Kolesnikov and Kumaresan (2013), $P_0$ and $P_1$ stand for sender and receiver.}
\begin{itemize}
\item \textbf{INPUT} of $P_0$: $m$ tuples $(x_j,0,...,x_{j-n})$ of $\ell$-bit messages, $1 \leq j \leq m$.
\item \textbf{INPUT} of $P_1$: $m$ selection integer $r = \langle r_1,...,r_m \rangle$ such that $0 \leq r_j < n$ for $1 \leq j \leq m$.
\item \textbf{COMMON INPUT}: A security parameter $\kappa$ such that $\kappa \geq n$, and Walsh–Hadamard codes $C_{\ell}^{\kappa} = \langle c_0,...,c_{\ell-1} \rangle$.
\item \textbf{ORACLE}: A random oracle $H : \langle m \rangle \times \langle 0,1 \rangle^\ell \rightarrow \langle 0,1 \rangle^\ell$.
\item \textbf{CRYPTOGRAPHIC PRIMITIVE}: A base OT$_m$ primitive.
\end{itemize}
\begin{enumerate}
\item $P_0$ chooses $s \leftarrow \langle 0,1 \rangle^\kappa$ at random. Let $s_i$ denote the $i$th bit of $s$.
\item $P_1$ forms $m \times \kappa$ matrices $T_{01},T_1$ in the following way:
\begin{itemize}
\item Choose $t_{j,0},t_{j,1} \leftarrow \langle 0,1 \rangle^\kappa$ at random such that $t_{j,0} \oplus t_{j,1} = c_{r_j}$
\end{itemize}
Let $t_{j,0}$, $t_{j,1}$ denote the $i$th column of matrices $T_0,T_1$, respectively.
\item $P_0$ and $P_1$ interact with OT$_m$ in the following way:
\begin{itemize}
\item $P_0$ acts as receiver with input $[s_i]_{i \in [\kappa]}$.
\item $P_1$ as sender with input $[t_{i,0},t_{i,1}]_{i \in [\kappa]}$.
\item $P_0$ receives output $[q]_{i \in [\kappa]}$.
\end{itemize}
\end{enumerate}
\end{algorithm}

2.2. 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 $[cx]$.

There are many public-key cryptosystems that satisfy this condition. However, related works mostly used the Paillier cryptosystem (Pailler, 1999). In this scheme, the public key is the product of two private prime numbers, $p$ and $q$, and $\phi$ denotes its bit-length where bit-length of the cipher-text is $2 \times \phi$.

2.3. Oblivious transfer (OT)

Two parties, called the sender and the 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 $r \in \{0,\ldots,n-1\}$ to obtain $x_r$.

Preliminary OT-based protocols consist of expensive public-key operations while recent improvements of OT, called OT-extension (Asharov et al., 2013; Kolesnikov and Kumaresan, 2013), allow the extension of a small number of base OTs using only symmetric operations. For the base OTs, a constant number ($\kappa$) of public-key operations is required.

Two approaches to perform base OTs are described below. The first approach is based on HE while the second one uses the Diffie–Hellman (DH) key exchange protocol (Chou and Orlandi, 2015).

1-out-of-2 OT using HE: The receiver generates the 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 $B = A g^b$ (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 $k_b = H(A^b)$ and decrypts the desired message by its key as $x_r = D_{k_b}(e_r)$. $H$ 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. The DH key exchange protocol is secure due to the hardness of breaking the Diffie–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 Kolesnikov and Kumaresan (2013) 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 are executed $\kappa$
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 Kolesnikov and Kumaresan (2013, Section 4). The protocol’s complexity is explained below.

The protocol executes $OT^m_{2^k}$, which has a complexity equal to that of $OT^m_{2^k}$ (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 $OT^m_{m}$ corresponds to the communication complexity of $OT^m_{2^k}$ plus $mnl$ transferred bits between sender and receiver in Step 4, that is, $O(m(k+n))$ bits. Consequently, the total computation complexity of the protocol is proportional to its communication complexity.

### 2.4. Arithmetic sharing (AS)

Assuming distance functions consist of a series of addition and multiplication operations, we use the Arithmetic sharing concept to calculate the distance privately. Arithmetic sharing is used to address distance functions of the first type described in Section 1.

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 satisfy $x^0 + x^1 = x \mod Z_l$.

Given two shares $x^0$, $x^1$, it is possible to perform 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 the multiplication of $x$ and $y$ ($x \times y$). The Arithmetic multiplication protocols 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 the $Sh(x)$ 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 the $Rec(x)$ 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 $X = [x_1, \ldots, x_m]$ and $Y = [y_1, \ldots, y_m]$ 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 (Bai et al., 2014).

#### 3.2. Manhattan distance (MD)

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

#### 3.3. Mahalanobis distance (MHD)

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

#### 3.4. 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 $X = [x_1, \ldots, x_m]$ and $Y = [y_1, \ldots, y_m]$ is computed as $CS(X, Y) = \sum_{i=1}^{m} x_i \cdot y_i / |X||Y|$, where $|X|$ and $|Y|$ are the norm of the vectors $X$ and $Y$ respectively, i.e., $|X| = \sqrt{\sum_{i=1}^{m} x_i^2}$.

#### 3.5. Scalar product (SP)

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

#### 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 us 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–Fischer algorithm (Wagner and Fischer, 1974) is $O(m \times m')$ or $O(m^2)$. This quadratic complexity makes the algorithm inefficient, particularly in the cryptography domain.

Ukkonen’s algorithm (Hyyrő, 2003) improves upon the Wagner–Fischer 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 multiplication used in distance functions of the first type which consist of arithmetic operations and secure comparison that is the building block for Edit distance where the goal is to compare two discrete values.

4.1. Secure arithmetic multiplication

As shown in Sections 3.1 to 3.3, 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 Atallah et al. (2004). Then we summarize the OT-based protocol proposed in Gilboa (1999). Finally, we introduce our protocol which shows significant improvement over (Gilboa, 1999).

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 $P_1$ (Atallah et al., 2004). 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 (HE-based).

1. $P_0 : x \in \mathbb{Z}_2^l$
2. $P_1 : y \in \mathbb{Z}_2^l$, $r \in \mathbb{Z}_{2^{l+1}}$, $z^1 = -r$
3. $P_0 \rightarrow P_1 : [x]$
4. $P_1 \rightarrow P_0 : [d] = [x]^y \cdot [r]$
5. $P_0 : z^0 = d$

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

Security analysis: Because all messages received by $P_1$ are encrypted under the 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 $[d] = [x \cdot y + r]$ that are statistically indistinguishable from uniformly random values selected from $\mathbb{Z}_{2^{l+1}}$.

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 $2p$ bits, so the total communication is $4p$ 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 (Gilboa, 1999) 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.

Algorithm 3 Arithmetic multiplication using OT (OT-based).

1. $P_0 : x \in \mathbb{Z}_2^l$
2. Binary representation of $x : x_0, \ldots, x_0$
3. $P_1 : b \in \mathbb{Z}_2^l, s_0, \ldots, s_{l-1} \in \mathbb{Z}_2^l$

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 $t_i^b$ from the pair $(t_i^0, t_i^1)$. The correctness and security of this protocol have been proven in Gilboa (1999, Section 4.1).

Communication complexity: The complexity of the secure multiplication protocol depends on the underlying OT protocol. Using the OT-extension proposed in Kolesnikov and Kumaresan (2013), total communication is $l(k + 2l)$ 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 the 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 $\rho$ times, in our improved protocol 1-out-of-16 OT executes $\rho/4$ times. From now on, $\rho/4$ is denoted by $\rho'$. As shown in Algorithm 4, our modification proceeds by following these steps:

1. $P_1$ selects $\rho'$ random and independent elements denoted by $s_0, \ldots, s_{\rho'-1} \in \mathbb{Z}_2^l$. It then prepares $\rho'$ sets with sixteen elements in each: $(t_{0,0}^{\rho'}, \ldots, t_{0,15}^{\rho'}), (t_{1,0}^{\rho'}, \ldots, t_{1,15}^{\rho'}), \ldots, (t_{\rho'-1,0}^{\rho'}, \ldots, t_{\rho'-1,15}^{\rho'})$. For every $0 \leq i \leq \rho' - 1$ 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 $x_0, x_1, \ldots, x_0$, $P_0$ and $P_1$ execute 1-out-of-16 OT $l'$ times on $l$-bit messages. In the $i$th execution, $P_0$ chooses $t_i^{s_i}$ from the set $(t_0^{s_i}, \ldots, t_{15}^{s_i})$.
3. $P_0$ sets $z^0 = \sum_{i=0}^{l'-1} t_i^{s_i}$ and $P_1$ sets $z^1 = \sum_{i=0}^{l'-1} t_i^{s_i}$.

Correctness analysis: Since $x_0, x_1, \ldots, x_0$ is the hexadecimal representation of $x$, one can write $x = \sum_{i=0}^{l'-1} x_i \cdot 16^i$ as shown in Eq. (1):
Table 1 – Communication Complexity of Arithmetic multiplication protocols in terms of transferred bits between client and server.

<table>
<thead>
<tr>
<th>(\psi = 1024, \kappa = 80)</th>
<th>1</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>HE-based</td>
<td>4098</td>
<td>4098</td>
<td>4098</td>
<td>4098</td>
<td>4098</td>
</tr>
<tr>
<td>OT-based</td>
<td>768</td>
<td>1792</td>
<td>3072</td>
<td>8018</td>
<td>4608</td>
</tr>
<tr>
<td>Our Approach</td>
<td>416</td>
<td>1344</td>
<td>2784</td>
<td>4736</td>
<td>7000</td>
</tr>
<tr>
<td>Improvement over HE-based</td>
<td>90%</td>
<td>67%</td>
<td>32%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Improvement over OT-based</td>
<td>46%</td>
<td>25%</td>
<td>10%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(\psi = 1024, \kappa = 128)</td>
<td>1</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>HE-based</td>
<td>4098</td>
<td>4098</td>
<td>4098</td>
<td>4098</td>
<td>4098</td>
</tr>
<tr>
<td>OT-based</td>
<td>768</td>
<td>1792</td>
<td>3072</td>
<td>8018</td>
<td>4608</td>
</tr>
<tr>
<td>Our Approach</td>
<td>512</td>
<td>1536</td>
<td>3072</td>
<td>5120</td>
<td>7000</td>
</tr>
<tr>
<td>Improvement over HE-based</td>
<td>87%</td>
<td>62%</td>
<td>25%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Improvement over OT-based</td>
<td>55%</td>
<td>40%</td>
<td>27%</td>
<td>16%</td>
<td>–</td>
</tr>
<tr>
<td>(\psi = 2048, \kappa = 80)</td>
<td>1</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>HE-based</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
</tr>
<tr>
<td>OT-based</td>
<td>768</td>
<td>1792</td>
<td>3072</td>
<td>8018</td>
<td>4608</td>
</tr>
<tr>
<td>Our Approach</td>
<td>416</td>
<td>1344</td>
<td>2784</td>
<td>4736</td>
<td>7000</td>
</tr>
<tr>
<td>Improvement over HE-based</td>
<td>95%</td>
<td>83%</td>
<td>66%</td>
<td>42%</td>
<td>–</td>
</tr>
<tr>
<td>Improvement over OT-based</td>
<td>46%</td>
<td>25%</td>
<td>10%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(\psi = 2048, \kappa = 128)</td>
<td>1</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>HE-based</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
</tr>
<tr>
<td>OT-based</td>
<td>1152</td>
<td>2560</td>
<td>4224</td>
<td>6144</td>
<td>8192</td>
</tr>
<tr>
<td>Our Approach</td>
<td>512</td>
<td>1536</td>
<td>3072</td>
<td>5120</td>
<td>7000</td>
</tr>
<tr>
<td>Improvement over HE-based</td>
<td>94%</td>
<td>81%</td>
<td>62%</td>
<td>37%</td>
<td>–</td>
</tr>
<tr>
<td>Improvement over OT-based</td>
<td>55%</td>
<td>40%</td>
<td>27%</td>
<td>16%</td>
<td>–</td>
</tr>
</tbody>
</table>

\[
z^0 + z^1 = \sum_{i=0}^{\psi-1} i^\kappa - \sum_{i=0}^{\psi-1} s_i \\
= \sum_{i=0}^{\psi-1} (x_i \cdot 16^i y + s_i) - \sum_{i=0}^{\psi-1} s_i \\
= y \cdot \sum_{i=0}^{\psi-1} x_i \cdot 16^i \\
= x \cdot y
\]  

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 \(s_0, \ldots, s_{\psi-1}\) 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 Kolesnikov and Kumaresan (2013), total communication takes \(l(\kappa + 16l)\) per Arithmetic multiplication.

Note that \(\kappa\) 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 Algorithms 2 to 4 with respect to different values of the symmetric and public key security parameters \((\psi, \kappa)\) and different lengths 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. After increasing the lengths of messages, we can see as we choose stronger security parameters \((\psi = 2048, \kappa = 128)\), 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 \(X = \{x_0, \ldots, x_{M-1}\}\) and \(Y = \{y_0, \ldots, y_{M-1}\}\) 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:

\[
M_{k}(0:k(N-1)) = \begin{cases} 
1 & \text{if } (k - y_j) \mod N = 0 \\
0 & \text{else}
\end{cases}
\]  

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, execution of the 1-out-of-\(n\) protocol proposed in Kolesnikov and Kumaresan (2013) 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 \((M_{k})\). 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 IDA (2016) we found that these values for the threshold cause the algorithm to return a 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 \( k + N \) bits where \( N \) is database size. The Edit distance algorithm requires \( k \times m \) number of comparisons so the consumption of the bandwidth is \( \left( k \times m \right) \times (k + N) \) bits. Concretely, the value of threshold \( k \) impacts communication complexity linearly. In our experiments, we evaluate the privacy-preserving Edit distance algorithms with following values of \( k \): 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 times compared to C++ implementations of cryptographic protocols. Hence, we focus on the communication complexity and number of required computational operations and do not focus on the absolute value of execution time but we aim to evaluate the changes in execution time in the presence of databases with different sizes and messages with different bit-lengths.

5.1. Experimental setup

We evaluate our approach with respect to execution time and communication bandwidth. The goal of this 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 have an 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-bit numbers in the range \([0, 255]\). For Edit distance, we evaluate our protocol using a genome database released by “iDASH Security and Privacy Workshop” IDA (2016).

5.1.2. Security parameters

We evaluate our system with different public key security parameters \( \psi \in [1024, 2048] \). The symmetric security parameter \( \kappa \) is set to be 80 or 128. Although the related work that are based on 1-out-of-2 OT set security parameters as \( \psi = 1024 \) and \( \kappa = 80 \) (Bringer et al., 2014), we evaluate feasibility of our protocol in terms of both computation and communication in the presence of stronger security parameters by considering key security parameters with a longer bit-length that is \( \psi = 2048 \) 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:

\[
ED(X, Y) = \sum_{i=1}^{m} x_i^2 - 2 \times \sum_{i=1}^{m} (x_i \cdot y_i) + \sum_{i=1}^{m} y_i^2
\]

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 \( D^0 = \sum_{i=1}^{m} (x_i^2 - 2x_i y_i) \) and the server obtains the other share of the distance as \( D^1 = \sum_{i=1}^{m} (y_i^2 - 2z_i) \).

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 the privacy-preserving approach will not affect the accuracy of the results and the private distance is equal to the 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.

Figs. 2 and 3 show the execution time and bandwidth respectively when \( \psi = 1024 \) and \( \kappa = 80 \) while the length of OT messages changes from 8-bit to 32-bit in the presence of different database sizes: 128, 256, 320 and 512 patterns. Note 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 s on the WAN network. This time also includes communication time which means if we execute our protocol on a LAN network it would take less time. The communication bandwidth varies from 190 KB to 600 KB in the presence of 8-bit long messages in length while database size varies from 128 to 512 (Fig. 3). As we can see in Fig. 2, the execution time does not scale proportionally when we double the size of the database. For example, the execution time is 3 s and 3.4 s in the presence of databases with 128 and 512 in size respectively when message length is 8 bits.

\(^2\) http://people.tamu.edu/~kagbazaran/Privacy_Preserving_Distance_Calculation_Framework.7z.
Figs. 4 and 5 show 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 identification protocol takes only 4.2 s on the WAN network. The communication bandwidth varies from 200 KB to 800 KB in the presence of 8-bit long messages while database size varies from 128 to 512 (Fig. 5).

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.

### Table 2 – Performance evaluation of privacy-preserving Euclidean distance when $\varphi = 2048$.  

<table>
<thead>
<tr>
<th>$\kappa$</th>
<th>Execution time (s)</th>
<th>Communication bandwidth (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>80</td>
<td>10</td>
<td>9.1</td>
</tr>
<tr>
<td>128</td>
<td>12.2</td>
<td>13.6</td>
</tr>
</tbody>
</table>

### 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 $CS(X, Y) = X \cdot Y$ where $|X|$ and $|Y|$ 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 $CS(X, Y) = \sum_{i=0}^{m-1} x_i' y_i'$ where $x_i'$ and $y_i'$ are $\frac{x_i}{|X|}$ and $\frac{y_i}{|Y|}$, 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 a dot product problem, 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 the 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 $[0, 1]$, so
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 ac-
curacy 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 size 128, $\kappa = 80$ and $\psi = 1024$
as shown in Table 3. We can see that when the scaling factor
is 1000 we can achieve the best trade-off between accuracy
and performance i.e., reasonable error rate and communication
overhead. Eq. (4) shows how the error rate is calculated.
The error rate evaluates the impact of the Scaling process on
accuracy.

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

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.

Figs. 6 and 7 show the execution time and bandwidth re-
spectively when $\psi = 1024$ and $\kappa = 80$. For example, when
the size of the database is 256 and the bit-length of the OT
messages is 8 bits, the privacy-preserving identification protocol
takes only 4.5 s on the WAN network, while the communica-
tion bandwidth is 500 KB.

Figs. 8 and 9 show the execution time and bandwidth re-
spectively when $\psi = 1024$ and $\kappa = 128$. For example, when
the size of the database is 256 and the bit-length of the OT
messages is 8 bits, the privacy-preserving identification protocol
takes only 4.8 s on the WAN network, while the commu-
nication bandwidth is 600 KB.

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

<table>
<thead>
<tr>
<th>Table 3 – Cosine similarity accuracy analysis.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>10,000</td>
</tr>
</tbody>
</table>

Fig. 6 – Privacy-preserving cosine similarity execution time
when $\psi = 1024, \kappa = 80$.

Fig. 7 – Privacy-preserving cosine similarity bandwidth
when $\psi = 1024, \kappa = 80$.

Fig. 8 – Privacy-preserving cosine similarity execution time
when $\psi = 1024, \kappa = 128$. 
5.4. Privacy-preserving edit distance

To calculate the 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.

5.4.1. Performance evaluation of privacy-preserving edit distance

To evaluate our approach, we use the database of genome data IDA (2016) in which the server holds 50 different sequences. The length of the sequences in average is 3500 characters from the $[A, C, G, T]$ alphabet set. The experimental results are shown in Figs. 10 and 11 with different security parameters.

We also run the our Edit distance framework over a LAN network in addition to a WAN network. In the 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 (Al Aziz and et al., 2017) which run experiments on the same VM machines. Fig. 4 demonstrates the execution time over the LAN network.

Results: We set the 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 Figs. 10 to 12 with different security parameters ($\varphi \in \{1024, 2048\}$, $\kappa \in 80, 128$). Figs. 11 and 12 measure the running time in seconds on the LAN and WAN networks respectively while Fig. 10 shows the bandwidth consumption in KB. Execution time varies from 8 to 38 s on LAN and 45 to 75 s on WAN. The execution time on WAN is higher due to network latency. Bandwidth directly depends on the symmetric security parameter $\kappa$ or number of base-OTs, as shown in Fig. 10, the public-key security parameter $\varphi$ does not affect the communication.

Analysis: The private Edit distance protocols proposed in Al Aziz and et al. (2017) is executed in 23 s 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 s with same configuration. The other advantage of our approach over Al Aziz and et al. (2017) is that it calculates accurate Edit distance while the other work approximates the Edit distance.

5.5. Complexity analysis

In this section, we discuss the impact of database size and message bit-length on the protocol complexity in terms of both communication and computation time.

Communication bandwidth depends on three parameters (i) message bit-length, (ii) database size, and (iii) security parameters. However, database size does not affect the communication cost significantly while increasing message length from 8-bit to 32-bit causes a drastic increase in communication cost. This is due to the fact that the 1-out-of-n OT protocol
proposed in Kolesnikov and Kumaresan (2013) and adapted in our approach performs efficiently over short-length messages. Our experimental results also confirm that communication bandwidth directly depends on message bit-length and it is highly efficient in the presence of short messages.

As shown in Fig. 3, in the presence of 8-bit messages and security parameters \( \psi = 1024 \) and \( \kappa = 80 \), the increase in database size would lead to a not even linear increase in communication cost i.e., communication bandwidth to compute privacy-preserving Euclidean distance changes from 200 KB to 600 KB when database size changes from 128 to 512 (four times increase). This is the cost of distance calculation in the one versus many scenario. To investigate the scalability of our proposed privacy-preserving protocol in many versus many scenarios like verification use-cases where the frequency of calling on the distance calculation function is \( O(N^2) \) and \( N \) is the number of biometrics samples, we only need to assume that size of the database is increased by a factor of \( N \) in one versus many scenario. Hence, we will get a less than \( N \) times increase in the communication bandwidth as it does not increase linearly by database size.

Table 1 also provides more support that communication cost is highly influenced by message bit-length such that our proposed OT-base multiplication gain about 90% and 46% improvement over Homomorphic and 1-out-of-2 OT based multiplications respectively for 8-bit messages while the improvement is not equally significant as bit-length increases though there is still promising improvement e.g., 32% improvement over Homomorphic encryption for 24-bit messages.

In summary, our proposed protocol performs efficiently in the presence of short-length messages and database size does not impact the communication bandwidth quadratically.

Coming up to computation cost analysis, it is measured by execution time. As we can see from Figs. 2, 4, 6 and 8, execution time does not go up even linearly but a negligible increase happens when we expand the database size. For example, in the presence of 8-bit messages execution time varies from 3 to 4.1 s when database size varies from 128 to 512 for privacy-preserving Euclidean distance – the same pattern is observable in cosine similarity setting.

It should be noted that reported execution time includes both computation and transferring time. Obviously, when it runs over a LAN network, it executes significantly faster as a big portion of the spent time is due to network latency. Edit distance is evaluated over both LAN and WAN networks. For example, execution time is 8 s over LAN while it is 45 s over WAN in the same settings of security parameters and distance threshold. It gives us a strong support for the idea that most of the execution time is spent due to network latency.

6. Related work

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

Euclidean distance: Most 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 (Erkin et al., 2009). 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 (Sadeghi et al., 2009). Barni et al. proposed a privacy-preserving protocol for fingerprint identification using the Paillier Homomorphic encryption (Barni et al., 2010). Evans et al. proposed a privacy-preserving biometric identification protocol with a focus on fingerprint data (Evans et al., 2011). This protocol optimizes the complexity of the Paillier cryptosystem by proposing a new technique for packing. Blanton and Gasti (2011) used DGK homomorphic cryptosystem for fingerprint identification. Chun et al. proposed a protocol in which data and computations are outsourced to a cloud (Chun et al., 2012). The server’s database and the client’s input are encrypted first 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 the underlying homomorphic encryption.

Cosine similarity: To our knowledge, there exist two works addressing privacy-preserving cosine similarity using homomorphic encryption (Kikuchi et al., 2010; Yang et al., 2013). The proposed approach in Yang et al. (2013) 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 Kikuchi et al. (2010) 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: Shantau and Boufounos proposed an approach to calculate Edit distance using HE (Aguilar-Melchor et al., 2013). They reduced the problem to a 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 (Huang et al., 2011).
In a related direction the survey article described in Niksefat et al. (2017) reviews privacy preserving techniques.

7. Conclusion and future work

In this paper, we develop a general framework to address privacy-preserving distance calculation in an 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 privacy-preserving Edit distance using OT for the first time.

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 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 the 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 the minimum distance is returned to the client (Bringer et al., 2014; Erkin et al., 2009; Evans et al., 2011; Kaghazgaran and Sadeghyan, 2011; Lazzeretti and Barni, 2013; Sadeghi et al., 2009).
2. All the identities with distance values less than a specific threshold are returned to the client (Barni et al., 2010; Chun et al., 2014; Osadchy et al., 2010).
3. The client learns the comparison results between its input and all the records in the server's database (Blanton and Gasti, 2011).

There are some works that use different techniques for comparison such as Homomorphic encryption (Barni et al., 2010; Erkin et al., 2009), Garbled circuit (Blanton and Gasti, 2011; Chun et al., 2014; Evans et al., 2011; Lazzeretti and Barni, 2013; Sadeghi et al., 2009) and GMW (Bringer et al., 2014).

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 specific threshold using existing protocols. 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.

In our future study, we will consider the malicious adversary model by proposing our protocols based on committed Oblivious Transfer.

Conflict of interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cose.2019.01.010.

References

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/deep learning since 2016. She has been a research intern at Microsoft Research (MSR) in summer 2018. She is student member of IEEE and ACM.
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.
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.