A Survey on Secure Processing of Similarity Queries

Bin Mu

May, 2014

The Graduate Center of The City University of New York

bmu@gc.cuny.edu
Abstract

With the rapid growth of the volume and diversity of digital data produced by all kinds of commercial, scientific and leisure-time applications, the extraction of useful information from these large data sets has become one of the key IT tasks. On the other hand, the constant monitoring of people’s activities, while using these applications, has raised people’s concern about the invasiveness of their privacy. Furthermore, institutes that own sensitive databases want to keep their data private, while providing data mining services to other parties. Thus, a considerable amount of research effort has been invested in the secure processing of a large variety of queries. In this survey, I will review protocols that can process secure similarity queries in different application domains: location proximity detection, biometric data recognition, similar document detection, and search over multimedia database. Most of these protocols are based on the secure multiparty computation (SMC) model and are provably secure.
# Table of Contents

**List of Figures** ii  

1 Introduction 1  

2 Tools on Secure Computation 5  

2.1 Secure multiparty computation 5  

2.2 Homomorphic encryption 6  

2.3 Oblivious Transfer 7  

2.4 Garbled circuits 7  

3 Secure location proximity detection 9  

4 Secure biometric data recognition 12  

4.1 Secure face recognition 13  

4.2 Other biometric data recognition 20  

5 Secure similar document detection 24  

6 Secure search over multimedia database 28  

6.1 Secure image search 28  

6.2 Secure audio search 31  

7 Conclusion and future work 34
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gird distances in the Pierre protocol. Alice can only determine whether Bob is in the dark grey, medium grey, or light grey area.</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>The $F_{\text{threshold}}$ protocol</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Fingercode representation of fingerprints</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>The secure similar document detection protocol</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>The image retrieval process</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>The server database structure</td>
<td>31</td>
</tr>
</tbody>
</table>
1 Introduction

With the rapid growth of the volume and diversity of digital data produced by all kinds of commercial, scientific and leisure-time applications, the extraction of useful information from these large data sets has become one of the key IT tasks. Such data often does not provide sufficient meta-data description, therefore, in many applications, similarity search is more important than an exact match or keyword search. On the other hand, the constant monitoring of people’s activities, while using these applications, has raised people’s concern about the invasiveness of their privacy. Thus, a considerable amount of research effort has been invested in the secure processing of queries, especially similarity queries that match people’s interest.

The first topic concerns location similarity or proximity. The emergence of Geo-Social Networks (GeoSNs), such as Foursquare\(^1\) and Loop\(^2\), facilitates the development of novel applications that combine social networking features with location based services. In particular, a GeoSN enhances the traditional social networking graph with spatial information, by allowing users to “check in” at arbitrary geographic locations. Mobile users may then utilize this information to identify their social friends that are spatially close (e.g., in order to meet at a nearby coffee shop). This is typically called a proximity detection query.

A trivial way to process such queries is to store all location information at the GeoSN server, in plaintext format. Alternatively, users may choose to bypass the server and exchange their locations (in plaintext), on-demand, in a peer-to-peer manner. Clearly, both methods might reveal a lot of information about an individual’s lifestyle to the GeoSN server and/or his friends. If the leaked information is more than what the user is willing to disclose, he may be discouraged from registering with the GeoSN. Therefore, to protect privacy, GeoSN queries should not disclose any additional information regarding the location of a

\(^{1}\)https://foursquare.com/
\(^{2}\)https://www.loopt.com/
user, besides the information that can be derived from the query result. To this end, private proximity detection protocols allow any two parties to test for proximity while maintaining their locations secret. In particular, a private proximity detection query returns only a boolean result to the querier and, in addition, it guarantees that no party can derive any information regarding the other party’s location.

Secure biometric recognition is another topic that attracts people’s attention. Biometric techniques have advanced over the past years to a reliable means of authentication, which are increasingly deployed in various application domains. As a matter of fact, biometric templates are uniquely associated with each user and thus provide the credibility of the user. For the same reason, the possibility that biometric data could be leaked or exchanged raises concerns on its uses and abuses. For example, a government agency or an institute which maintains personal data might monitor and track the actions and the behavior of each individual. Hence the widespread use of biometric systems requires the protection of the biometrics templates and the protection of user privacy as well. It is even more important to note that the biometric matching process may involve a central server or be adopted in partially untrusted environments. The server that processes the biometric matching should not learn anything on the database and it should be impossible for the server to exploit the resulting matching values in order to extract any knowledge about the user presence or behavior.

Similar document or file detection is a useful technology in many applications. For example, detecting similar files facilitates better indexing, and provides efficient access to the file system that clusters similar files together. Furthermore, it can detect security breaches by identifying file versions that are modified by a virus or a hacker. Similar file detection can also help detect copyright violations and plagiarism. If a newly published paper is suspected of plagiarism, similar file detection can be used to determine whether this paper is similar to some paper in a database. While plaintext similar document detection is extremely
important, it is insufficient when the situation requires that the privacy of the contents of documents need to be maintained. For example, a health monitoring agency may want to know if what it has recorded is similar to reports at other agencies. Due to privacy and confidentiality issues, the agencies may be reluctant to share freely their reports. Therefore, traditional similar file detection techniques are not applicable here. Keeping academic integrity is another area where detection of similar confidential documents may be needed. For example, a conference’s organization committee may want to know whether the articles submitted to their conference are also submitted to other publication venues. Since research papers are regarded confidential until published, conferences need to compare submissions in a privacy-preserving way. To solve the above problems, people propose secure similar document detection (SSDD) protocol [18, 27, 19, 5] that can provide similar detection service in a privacy-preserving way.

Content-based multimedia (images, videos, audio clips) searches are growing to be a key component in many emerging applications. For example, people may use the Internet search engine to find images similar to an uploaded image. While ordinary users may not care about the leakage of their query to the search engine server, some users do want to keep the privacy of their query. An example is the logo patent search. A company would like to know if its new logo is very similar to any of the existing logos to avoid copyright infringements. However, for the fear of the leakage of originality in their newly designed logo, the company would not like to reveal it to the database that stores all logos that have already been registered. In the audio search area, speech recognition service is an edifying example. When users uses a speech recognition system, they must grant the system complete access to their voice recordings, which may lead to abuse of user’s personal information. For example, the system could infer information such as gender, ethnicity from the recording. Thus secure speech matching is getting more and more attention nowadays [36, 34, 35].

The rest of this survey is organized as follows: In Section 2 I will briefly introduce the
tools that are usually employed in building secure protocols. In Section 3 I will survey research papers on secure location proximity detection. In Section 4 I will survey research papers on secure biometric data recognition. In Section 5 I will survey research papers on secure similar document detection. In Section 6 I will survey research papers on secure multimedia search. Finally, I will conclude this work and discuss my future research directions in Section 7.
2 Tools on Secure Computation

2.1 Secure multiparty computation

The word secure is related to the definition of Secure Multiparty Computation (SMC). Yao’s Millionaires’ problem is probably the first secure two-party computation problem in the literature[49]. The problem discusses two millionaires, Alice and Bob, who are interested in knowing which of them is richer without revealing their actual wealth. Yao developed a provably secure protocol that solves this problem. In general, a secure two-party computation protocol enables two parties to joint compute a function based on their respective inputs, without having to reveal their input to the other party. This was extended to multi-party computation by Goldreich et al[13], who also developed a general framework for SMC.

There are two standard adversarial models under SMC: semi-honest (also known as honest-but curious or passive) and malicious. A semi-honest party follows the prescribed behavior, but might try to compute additional information from the information obtained during protocol execution. What a party knows during an execute of the protocol is defined as the party’s view. Security in this setting is defined using simulation argument: the protocol is secure if the view of the protocol execution for each party is computationally indistinguishable from the view simulated using that party’s input and output only. This model guarantees that parties that correctly follow the protocol cannot gain any knowledge about the other party’s input except the output and whatever can be inferred from its own input. The definition below formalizes the notion of security for semi-honest participants[12]:

**Definition 1** Let \( f : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^* \) be a functionality and \( f_1(x, y) \) and \( f_2(x, y) \) denote the first and the second element of \( f(x, y) \) respectively. Let \( \Pi \) be two-party protocol for computing \( f \). The views of the first and the second parties during an execution of \( \Pi \) on \( (x, y) \), denoted by \( \text{VIEW}_1^\Pi(x, y) \) and \( \text{VIEW}_2^\Pi(x, y) \), are \( (x, r, m_1, \ldots, m_t) \) and \( (y, r, m_1, \ldots, m_T) \) respectively, where \( r \) represents the outcome of the first and second
parties’ internal coin tosses, and $m_i$ represents the $i^{th}$ message they have received. The outputs of the first and the second parties during and execution of $\Pi$ on $(x, y)$, denoted by $\text{OUTPUT}^\Pi_1(x, y)$ and $\text{OUTPUT}^\Pi_2(x, y)$, are implicitly included in the party’s own view of the execution, and $\text{OUTPUT}^\Pi(x, y) = (\text{OUTPUT}^\Pi_1(x, y), \text{OUTPUT}^\Pi_2(x, y))$. We say that $\Pi$ privately computes $f$ if there exist probabilistic polynomial-time algorithms (simulators), denoted $S_1$ and $S_2$, such that

\[
\{(S_1(x, f_1(x, y)), f(x, y))\}_{x, y} \equiv C \{(\text{VIEW}^\Pi_1(x, y), \text{OUTPUT}^\Pi_1(x, y))\}_{x, y}
\]

\[
\{(S_2(x, f_2(x, y)), f(x, y))\}_{x, y} \equiv C \{(\text{VIEW}^\Pi_2(x, y), \text{OUTPUT}^\Pi_2(x, y))\}_{x, y}
\]

where $\equiv$ denotes computational indistinguishability by families of polynomial-size circuits.

However, a malicious adversary is allowed to deviate arbitrarily from the specified protocol. It adds the complexity of the problem since there are certain undesirable actions that cannot be prevented. While many of the existing practical SMC protocols do provide certain guarantees beyond that of the semi-honest model, few meet all the standards of the malicious model. So only protocols on semi-honest model will be discussed in the following sections.

### 2.2 Homomorphic encryption

Homomorphism in encryption allows one to evaluate arithmetic operations, such as multiplication and addition, over plaintext values by manipulating their corresponding ciphertexts. Most public key cryptosystems in the literature are partially homomorphic, i.e., they facilitate the evaluation of one algebraic operation (either addition or multiplication). In general, additively homomorphic encryption schemes are more often used in the literature. The
properties of this kind of cryptosystems are as follows (where $E(\cdot)$ denotes encryption):

$$E(m_1 + m_2) = E(m_1)E(m_2)$$

$$E(m_1 - m_2) = E(m_1)E(m_2)^{-1}$$

$$E(m \cdot k) = E(m)^k$$

Examples of this kind of cryptosystems are Paillier[33] and ElGamal[8] cryptosystems. Both cryptosystems are semantically secure, i.e., it is infeasible to derive any information about a plaintext, given its ciphertext and the public key that was used to encrypt it. The security of Paillier’s scheme is based on the decisional composite residuosity assumption, while the security of ElGamal’s scheme is based on the decisional Diffie-Hellman assumption.

2.3 Oblivious Transfer

An oblivious transfer protocol allows a sender to send one of a possible set of values to a receiver; the receive selects and learns only one of the values, and the sender does not learn which value the receiver has selected. In 1-out-of-2 oblivious transfer, denoted $OT^2_1$, one party, the sender, has as its input two messages $m_0$, $m_1$ and another party, the receiver, has as its input a bit $b$. At the end of the protocol, the receiver learns $m_b$ and the sender learns nothing. Similarly, in 1-out-of-N OT the receiver obtains one of the $N$ messages held by the sender. There is a rich body of research literature on OT protocols, such as [28][37][23].

2.4 Garbled circuits

Originated in Yao’s work[19], garbled circuit allows two parties to securely evaluate any function represented as a boolean circuit. Lindell and Pinkas[22] provide a full description and complete proof of security. The idea is for one party (the circuit generator) to represent
boolean wire values on each wire with a cryptographic key called that wire’s *wire label*, and to replace each gate’s truth table with a corresponding garbled gate. Garbled gates are constructed by encrypting outgoing wire labels for each gate using an appropriate combination of the two input wire labels. The second party (the *circuit evaluator*) obtains the input wire labels using oblivious transfer, then the evaluator can evaluate the circuit without any further interaction. For each garbled gate, the evaluator can decrypt exactly one entry for the outgoing wire based on the wire labels he/she knows for the two input wires. The circuit generator also sends the mapping from wire labels to boolean values for the output wire, so the result of the computation can be recovered by linking the output wire label to the corresponding boolean value.
3 Secure location proximity detection

Ruppel et al. [39] utilize a symmetric key cipher that encrypts locations by applying a distance-preserving transformation. A set of friends share a common key and use it to encrypt their location prior to uploading it to the server. Due to the distance-preserving property of the transformation, the server can determine whether any two friends are within a given proximity threshold. Clearly, this approach leaks some location information, as the server learns the actual distances among all users. Furthermore, if a user colludes with the server and reveals the shared key, all user locations are compromised. Longitude [25] is a similar approach, but the underlying transformation does not disclose the exact distances (i.e., it results in a loss of accuracy).

Most private proximity detection algorithms in the literature employ a tessellation method (typically a regular grid) to partition the space into a fixed number of cells. In this way, they reduce the proximity detection problem into an equality testing problem: identify whether the two parties are located inside the same or nearby cells. FriendLocator [45] and VicinityLocator [44] assume that the two parties share a secret key and use it to encrypt (with a deterministic symmetric cipher) the ids of certain cells nearby their location. The encrypted values are uploaded to server, who can determine (by matching the ciphertexts) whether the two users lie in the same or adjacent cells of the grid. Clearly, both schemes are vulnerable to collusions with the server, since the party that colludes can learn the approximate location of the other party.

In C-Hide&Seek [26], every user shares his secret key with his friends, and uses this key to encrypt his up-to-date location. When another user issues a proximity request, the server simply forwards all the encrypted locations that it currently stores. Therefore, the proximity detection is performed at the querier, which enables him to identify (with a simple brute force approach) the approximate locations of all his friends. C-Hide&Hash [26] assumes a
similar location update procedure as *C-Hide&Seek*. However, for proximity detection, it employs a secure computation protocol between the server and the querier. Nevertheless, due to the shared keys among the users, this scheme is also vulnerable to collusions with the server.

Zhong et al. [50] propose three schemes, namely *Louis*, *Lester* and *Pierre* that are based on secure computations. The main idea in all protocols is to compute the distance between two parties using the properties of homomorphic encryption. First, *Louis* computes the actual distance, but requires a trusted third-party that only returns the result (i.e., true or false) of the proximity detection query. *Lester* does not require a third-party, but instead masks the actual distance $d$ in a way that its computation time increases linearly with $d$ (i.e., $d$ is retrieved efficiently only when its value is relatively small). Finally, *Pierre* utilizes a regular grid to discretize the users’ locations. It then employs a secure two-party computation protocol to determine whether the users lie within the same or adjacent cells in the grid. The resolution distance of the grid is $r$. Alice and Bob then compute their coordinates in units of $r$. If Alice’s real position is $(x, y)$, then her coordinate in the grid is $(x_r, y_r) = ([\frac{x}{r}, \frac{y}{r}])$. Similarly Bob’s coordinate in the grid is $(u_r, v_r)$. The protocol uses the properties of homomorphic encryptions. To start a query, Alice sends to Bob $r$, $E(x^2r + y^2r)$, $E(2xr)$, $E(2yr)$. Bob picks three random elements $\rho_0, \rho_1, \rho_2$ in the plaintext space and replies with $E(\rho_0 \cdot D_r), E(\rho_1 \cdot (D_r - 1)), E(\rho_2 \cdot (D_r - 2))$, where $D_r = (x_r - u_r)^2 + (y_r - v_r)^2$ is the square of the Euclidean distance between Alice and Bob in the grid. Note that if Alice and Bob are in the same grid cell, $\rho_0 \cdot D_r$ is 0; otherwise, $\rho_0 \cdot D_r$ is a random element in the plaintext space. Similarly, $\rho_1 \cdot (D_r - 1)$ is 0 if Alice and Bob are in adjacent grid cells and random otherwise, and $\rho_2 \cdot (D_r - 2)$ is 0 if Alice and Bob are in diagonally touching grid cells and random otherwise. See Figure 1. The paper shows that it is easy for Alice to check whether a receive ciphertext is an encryption of 0. Therefore, with this protocol, Alice can know whether Bob is in the same, adjacent, or diagonally touching grid cell of hers and
learns nothing else. Due to semantic security of the encryption, Bob cannot learn Alice’s location information.

Figure 1: Grid distances in the Pierre protocol. Alice can only determine whether Bob is in the dark grey, medium grey, or light grey area. [50]

Narayanan et al. [29] partition the space with three overlapping tessellations, in order to improve the accuracy of the proximity detection. When two parties want to test for proximity, they employ a secure computation protocol (similar to Pierre) to identify whether they are located in the same cell of at least one tessellation.

Among all the aforementioned protocols, Pierre [50] and Narayanan et al. [29] provide the strongest privacy guarantees, i.e., the querier only learns the proximity result, while all remaining parties learn nothing.
4 Secure biometric data recognition

Biometric data is more and more exploited for authentication and identification tasks in a multitude of applications ranging from institutional, medical, policy and commercial systems. Large-scale collections of biometric data include face, fingerprint and iris images collected by the US Department of Homeland Security (DHS) from travelers through its US-VISIT program[3]. While biometry serves as an good mechanism for authentication and identification of individuals, such data is extremely sensitive and need be well protected. Furthermore, once leaked biometric data cannot be revoked or replaced. For these reasons, biometric data cannot be easily shared between organizations or governments. However, there could be legitimate reasons to carry out computations on biometric data belonging to different entities. To this end, many research works focusing on privacy-preserving biometric identification are proposed[9, 32, 41]. The systems proposed in those works typically consist of a server-side database that holds a set of biometric records, and clients who submit a candidate biometric data to the server for identification or authentication. A common structure in those works has three main steps to be accomplished by the two parties[2]:

- feature extraction: The raw biometric data owned by both parties need to be converted, using some feature extraction methods, to feature vectors. There are mainly two reasons for this step. Firstly, it is usually very difficult or time-consuming to compare the raw biometric data in the matching step. Secondly, it is easier to process the extracted features in a privacy-preserving way.

- distances computation: the distances between the client vector and the vectors in the server’s database are computed in the ciphertext domain. Therefore the server cannot learn the actual value of the distance.

- selection of the matching identities: in this final step the server interacts with the client
in order to select, in the ciphertext domain, the enrolled record that has a minimum distance or records with the distances that are below a known threshold.

4.1 Secure face recognition

The increasing deployment of surveillance cameras in public places sparked interest in the use of face recognition technologies to automatically match faces of people shown on surveillance images against a database of known suspects. Erkin [9] developed a privacy preserving face recognition system based on the standard Eigenfaces recognition algorithm [46, 47]. This algorithm achieves reasonable classification rates of approximately 96%. The privacy preserving system works in the two-party setting in the semi-honest attacker model. For practical applications it is also assumed that both parties are computational bounded. Suppose the two parties are Alice and Bob. Alice owns a face image and wants to know whether her image shows a person in Bob’s database which contains a collection of face images from individuals. During the system execution, Alice will learn basic parameters of the face recognition system (such as the size of the database), but the content of Bob’s database will be kept private from Alice. Meanwhile, to keep Alice’s privacy, Bob will not learn her input image or the result of the recognition process.

In the feature extraction phase, the Eigenfaces algorithm transforms face images into characteristic feature vectors of a low-dimensional vector space, whose basis is composed of eigenfaces. From a collection of training images, the eigenfaces are computed through Principal Component Analysis (PCA). Every face image is succinctly represented as vector in the face space by projecting the face image onto the subspace spanned by the eigenfaces. A more detail description of the feature extraction can be found in [9]. To encrypt the features with Paillier’s cryptosystem and process them using homomorphic operations, the feature vectors are represented as integers. This is achieved by appropriate quantization: non-integer values are first scaled by a fixed scale factor and rounded to the nearest integer.
Each feature vector in the database owned by Bob is also associated with a string $Id_i$ that contains the identity of the person the feature vector belongs to. Suppose Bob’s feature vectors are $\Omega_1, \ldots, \Omega_M$. The size of the database is $M$.

Before the interaction starts, Alice generates a pair of public keys and private keys and sends her public key to Bob over an authenticated channel. During the interactive recognition protocol, Alice sends her encrypted face image $E(\Gamma)$ to Bob. Using the homomorphic properties and formulas described in [9], Bob obtains an encrypted feature vector of the face image as $E(\overline{\Omega}) := (E(\overline{\omega}_1), \ldots, E(\overline{\omega}_K))^T$. Then encryptions of the distances between $\overline{\Omega}$ and all feature vectors $\Omega \in \{\Omega_1, \ldots, \Omega_M\}$ from the database are computed. To simplify the computation, the paper uses square of the Euclidean distance as the metric:

$$D(\Omega, \overline{\Omega}) = \| \Omega - \overline{\Omega} \|^2 = \sum_{i=1}^{K} (\omega_i - \overline{\omega}_i)^2 = \sum_{i=1}^{K} \omega_i^2 + \sum_{i=1}^{K} (-2\omega_i \overline{\omega}_i) + \sum_{i=1}^{K} \overline{\omega}_i^2$$

Let $S_1 = \sum_{i=1}^{K} \omega_i^2$, $S_2 = \sum_{i=1}^{K} (-2\omega_i \overline{\omega}_i)$, $S_3 = \sum_{i=1}^{K} \overline{\omega}_i^2$. Remember that Bob knows the encryption of Alice’s feature vector $E(\overline{\Omega})$ and his feature vector $\Omega$ in the database. By the homomorphic property, $E(S_1) = E(\sum_{i=1}^{K} \omega_i^2) = \prod_{i=1}^{K} E(\omega_i^2)$, $E(S_2) = E(\sum_{i=1}^{K} (-2\omega_i \overline{\omega}_i)) = \prod_{i=1}^{K} E(\overline{\omega}_i)^{-2\omega_i}$. To obtain an encryption of $S_3$ requires more work. This is because Bob only knows the encryption of $\overline{\omega}_i$ and cannot perform the square without help from Alice. To prevent Alice from knowing $\overline{\omega}_i$, Bob additively blinds the value $\overline{\omega}_i$ with a uniformly random element $r_i$ from the plaintext space to obtain $E(x_i) = E(\overline{\omega}_i + r_i) = E(\overline{\omega}_i) \cdot E(r_i)$ and sends it to Alice who can decrypts. Then Alice computes the values $x_i^2$ in plain and obtains the value $S'_3 = \sum_{j=1}^{K} x_i^2$. She encrypts this value and sends it back to Bob, who can now compute $E(S_3)$ as follows:

$$E(S_3) = E\left(\sum_{i=1}^{K} \overline{\omega}_i^2\right) = E\left(\sum_{i=1}^{K} [(\overline{\omega}_i + r_i)^2 - 2r_i \overline{\omega}_i - r_i^2]\right) = E(S'_3) \cdot \prod_{j=1}^{K} (E(\overline{\omega}_i)^{-2r_i} \cdot E(-r_i^2))$$
Finally, Bob can compute the encryption of the distance:

\[ E(D(\Omega, \bar{\Omega})) = E(S_1) \cdot E(S_2) \cdot E(S_3) \]

In the match finding phase, the minimal distance and the corresponding identity \( Id \) must be found and returned to Alice. The paper uses a tournament method: in the first round, Bob compares the \( k = \lfloor \frac{M}{2} \rfloor \) encrypted distances \( E(D_{2i+1}) \) and \( E(D_{2i+2}) \) for \( 0 \leq i \leq k - 1 \), by using a secure cryptographic protocol that compares two encrypted values. The detailed comparison protocol can be found in [9]. To prevent Bob from determining the outcome of the comparison by inspecting the ciphertexts, Alice rerandomizes the encryption of the smaller distance and its \( Id \). At the end of this round, there will be \( \lceil \frac{M}{2} \rceil \) encryptions left. After \( \lceil \log_2(M) \rceil \) rounds there will only be one encryption left, the minimum. In order to check if the minimum distance is smaller than a threshold \( \tau \), one additional candidate with distance value \( \tau \) and a special identity 0 can be added to the tournament. After \( \lceil \log_2(M+1) \rceil \) rounds, Bob receives encryptions of the minimum distance and the corresponding identity. If a similar face is found in the database the identity is valid, otherwise the identity equals 0. Then the encryptions are sent to Alice who can decrypts.

A simple proof sketch of the security of the system is also provided in [9]. Firstly, the paper shows that the comparison protocol is secure. All messages received by Bob are encrypted under Alice’s public keys which cannot be decrypted by him. Alice on the other hand has access to the secret key. However, all messages received by Alice are masked by Bob, such that they are uniformly random over the whole plaintext space. Thus Alice can learn no useful information from the plaintexts of the encryptions. The proof security of the full protocol is similar to that of the comparison. In addition to the comparisons, interaction only happens when computing the distances. As above, the values \( x_1, \ldots, x_K \) that Alice receives are masked, so that Alice cannot learn any information about \( \bar{\omega}_i \). Bob again only
receives semantically secure encryptions, so he also learns nothing. Therefore, a formal
simulator proof can be easily constructed. Given one party’s input and output, simulation
of the other party is easy: Alice is handed encryptions of random values, while Bob can be
handed encryptions of 0, which are indistinguishable due to the semantic security.

Sadeghi[41] improved the efficiency of Erkin’s work by constructing a hybrid protocol
that uses homomorphic encryption to compute Euclidean distances and garbled circuits for
minimum finding. The minimum finding phase relies heavily on secure comparison. It is
shown in [20], the garbled circuit based comparison protocols clearly outperform protocols
based on homomorphic encryption. Moreover, [9] requires $O(\log M)$ rounds to find the most
similar face in a database of $M$ faces. Thus the communication complexity is high. So while
the distance computing part in [41] is the same as that in [9], [41] uses garbled circuits for the
minimum finding. The high-level structure of this part consists of several building blocks:
the sub-protocol ParallelConvert converts the homomorphically encrypted distances held by
the server $S$, $E(D_1), \ldots, E(D_M)$, into their corresponding garbled values $\tilde{D}_1, \ldots, \tilde{D}_M$ and
sends them to the client $C$. Then a garbled circuit $C_{\text{Minimum}}$ computes the minimum distance
from these garbled values. The garbled circuit $C_{\text{Minimum}}$ is created in the setup phase and
sent to $C$ before the online phase starts. The garbled value $\tilde{\tau}$ which correspond to server’s
threshold value $\tau$ is also sent to $C$. Finally, $C$ evaluates $C_{\text{Minimum}}$ on the garbled values
$\tilde{\tau}, \tilde{D}_1, \ldots, \tilde{D}_M$ and obtains the correct output $r$. An optimization method worth mentioning
is packing multiple values $D_i$ together into a singe ciphertext. It can reduce the number
of ciphertexts and save bandwidth. The basic idea is to send $M$ ciphertexts of the form
$E(D_i||D_{i+1}||\cdots||D_{i+l})$ instead of $M$ ciphertexts of the from $E(D_i)$, where the maximum
value of $l$ depends on the range of $D_i$ and the bit-length of plaintexts in the encryption
scheme being used.

SCiFI[32] is another practical privacy-preserving face identification system. It utilizes an
index-based face representation that is specifically designed for usage with secure computa-
tion. More specifically, the representation is based on the idea of facial composite, where a face is formed as a collection of fragments taken from vocabularies of facial features. It is analogous to the system used by police departments to record eyewitness’s memory of a face. The vocabularies of facial features consist of different typical appearances of facial parts extracted from a set of people unrelated to the face that should be reconstructed. It is assumed that the vocabulary contains a set of words (typical images) for each facial part, such as the nose, eyes, mouth, etc. Each part of the input image is assigned a set of words from the corresponding vocabulary, instead of a single match. As previous research shows that locations of facial features in a face have good discriminative power, the representation also takes these locations into account. In general, SCiFI represents a face by a vector which is composed of (1) indices into part vocabularies, and of (2) quantized relative distances of the parts to the center of the face. Any two representations are essentially sets that can be compared by a secure computation of their set difference.

Denote by \( N \) the number of words in each part vocabulary, and by \( p \) the number of parts in the face representation. A full face representation is in the format \( s = (s^a, s^s) \) and contains the following components:

- **Appearance component** \( s^a \): It is composed of \( p \) sets \( s^a_1, \ldots, s^a_p \), one set per facial part, where each set contains the indices of \( n \) out of \( N \) words of the part vocabulary. The \( n \) words that are most similar to the part \( i \) are selected for the set \( s^a_i \).

- **Spatial component** \( s^s \): It is composed of \( p \) sets \( s^s_1, \ldots, s^s_p \), where each set contains \( z \) indices of bins quantized distance from the center of the face. Denote the total number of these bins by \( Q \).

Moreover, SCiFI translates this representation to an equivalent representation as a binary vector, which is more convenient for applying the secure distance computation. Thus the set difference between the representations \( s, s' \) of two faces, is exactly equal to the Hamming
distance of the corresponding vectors. The binary vector representation is defined as follows:

- Every set $s_i^a$ is represented as a binary vector $v_i^a$ of length $N$. Each bit of $v_i^a$ corresponds to an index of any entry in the part vocabulary. In the locations of the $n$ indices in the set $s_i^a$, the corresponding bits of $v_i^a$ are set to 1, all other bits are set to 0.

- Every set $s_i^s$ is represented as a binary vector $v_i^s$ of length $Q$. Each bit of $v_i^s$ corresponds to a bin of quantized distances. In the locations of the indices in the set $s_i^s$, the corresponding bits of $v_i^s$ are set to 1, all other bits are set to 0.

The goal of the system is to know whether there is a match between the client’s input face image and a record in the database. SCiFI proposes two protocols which can be used for this purpose.

- $F_{\text{threshold}}$. In this protocol, there is an additional input, a threshold $t_i$, for each face in the server’s database. The output is the indices of the items in the database whose Hamming distance with the client’s input is smaller than the corresponding threshold $t_i$. The detailed protocol is shown in Figure 2.

- $F_{\text{min}+t}$. The output of this protocol is the index of the record in the database whose Hamming distance with the client’s input is minimal. If this distance is larger than the threshold, i.e., if there is no match found in the database, then no output is given.

The correctness of the $F_{\text{threshold}}$ protocol is also provided in the paper. By the properties of homomorphic encryption, the encryption of the Hamming distance between the client vector and the server vector is correctly computed. It is explained that with overwhelming probability the decryption of $E_{pk}(d_H + r_i)$ is equal to $d_H + r_i$ (i.e., there is no modular reduction). The client uses $(d_H + r_i) \mod d_{\text{max}} + 1$ as its input to the 1-outof-$(d_{\text{max}} + 1)$ OT protocol, while the server sets its input of the OT to be shifted $r_i$ locations. Therefore, the output of the client in the OT protocol is as required.
The following protocol uses a homomorphic cryptosystem. \( pk \) is a public key that both parties know, but only the client knows the private key. Both parties also know an upper bound \( d_{\text{max}} \) on the Hamming distance between the words.

**The client’s input**: a binary vector \( w = (w_0, \ldots, w_{l-1}) \);

**The server’s input**: a database of \( N \) binary vector, \( w^1, \ldots, w^N \), where \( w^i = (w^i_0, \ldots, w^i_{l-1}) \). The thresholds of each database record: \( t_1, \ldots, t_N \).

**Output**: the client learns the matching indices. The server learns nothing.

1. The client sends the homomorphic encryption of each bit of the binary vector, \( \{ E_{pk}(w_0), \ldots, E_{pk}(w_{l-1}) \} \), to the server.
2. For each bit location \( j \) the server calculates \( E_{pk}(w_j \oplus w^i_j) = E_{pk}(w_j + w^i_j - 2w_jw^i_j) = E_{pk}(w_j)E_{pk}(w^i_j)E_{pk}(w_j)^{-2w^i_j} \).
3. The server computes the encryption of the Hamming distance \( E_{pk}(d_H) = E_{pk}(\sum_{j=0}^{l-1} w_j \oplus w^i_j) = \prod_{j=0}^{l-1} E_{pk}(w_j \oplus w^i_j) \). The server chooses a random value \( r_i \), computes \( E_{pk}(d_H + r_i) \), and sends it to the client.
4. The client receives \( E_{pk}(d_H + r_i) \) and decrypts the result.
5. Using a OT\(_{d_{\text{max}}+1} \) protocol, the client get the matching result from the server:
   - The input of the client is \((d_H + r_i) \mod d_{\text{max}} + 1 \).
   - The inputs of the server are \( X_0, \ldots, X_{d_{\text{max}}} \), where
     \[
     X_j = \begin{cases} 
     1 & \text{if } 0 \leq (j - r_i) \mod d_{\text{max}} \leq t_i \\
     0 & \text{otherwise}
     \end{cases}
     \]

   ![Figure 2: The \( F_{\text{threshold}} \) protocol](image)

The security of the \( F_{\text{threshold}} \) is proved according to the simulation paradigm in the semi-honest model. More specifically, the paper shows that it is possible to simulate the view of each party given only the party’s input and output. The inputs of the client are the vector \( w \). The output consists of a bit for each record of the server. The simulator simulates the message received by the client in Step 4 with an encryption of a random value \( R \), which is regarded by the client as \( d_H + r_i \). Finally, the simulator sets the input of the client to the OT, which is \( R \mod (d_{\text{max}} + 1) \), and its output of the OT which is the corresponding bit of the server’s input. Consider the server. The simulator generates an encryption of \( l \) bits, all equal to 0, which are regarded by the server as the client’s input vector. Since the encryption scheme is semantically secure, the server cannot distinguish these encryptions.
from encryptions of the client’s actual vector. Then the server sends a message as in Step 4, and sets its input, which can be easily computed by the simulator, to the OT in Step 5.

For the $F_{\min+t}$ protocol, SCiFI\cite{32} suggests that it can be implemented using garbles circuits. The paper also presents an implementation based oblivious transfer. However, since the $F_{\text{threshold}}$ protocol outputs the identities of all database items which are close to client’s input, whereas $F_{\min+t}$ only outputs identity of the closest item, SCiFI\cite{32} argues that it is more reasonable to use $F_{\text{threshold}}$, since similarity to any of the suspects must be investigated.

### 4.2 Other biometric data recognition

Although face images are widely used in biometric-based identification services, they are known to be quite weak biometric traits. Therefore more reliable traits like fingerprint, iris code or DNA are likely to be used in applications that require higher security. Furthermore, the methods mentioned in the face recognition section are also applicable to these biometric data. As long as the features are extracted from the raw biometric data, the following procedures (distance computation, selection of the matching identities) are similar.

\cite{3, 2} propose protocols on secure fingerprint identification. In \cite{2}, the protocol is based on a particular representation of the fingerprints which yields a relatively short, fixed length code, called Fingerprint\cite{17} (illustrated in Figure 3). The fingerprint image is first divided into radial and angular sectors. The sectors are then filtered by a number of Gabor filters to determine the energy content of each sector into different frequency bands. Finally, the feature vector is formed by the sequence of a properly quantized version of sector energies.

The client’s input is a feature vector $\bar{x}$ to be identified. The server manages a database of $n$ pairs $(id^i, \bar{y}^i)$, where $id^i$ is a unique numeric identifier and $\bar{y}^i$ is the corresponding feature vector. The solution requires the identifiers be powers of 2 (i.e. $id^i = 2^i$). The distance computation part is very similar to that of Erkin\cite{9}. In the identities selection part, a more
efficient protocol called \( \text{bit} - \text{MIN} \) is designed to compare two encryptions. Once the distance computation phase is over, the server gets the encryptions of distances \( E(D^1), \ldots, E(D^n) \) and thus has the pairs: \((id^1, E(D^1)), \ldots, (id^n, E(D^n))\). The server permutes these pairs as: \((id^{j_1}, E(D^{j_1})), \ldots, (id^{j_n}, E(D^{j_n}))\) and then computes, using the protocol \( \text{bit} - \text{MIN} \), the values \( E(b^i) = \text{bit} - \text{MIN}(E(\tau), E(D^{j_i})) \) for \( i \in \{1, \ldots, n\} \). Finally, the server computes and returns to the client the following encrypted value:

\[
E(R) = E\left(\sum_{i=1}^{n} b^i \cdot id^i\right) = \prod_{i=1}^{n} E(b^i)^{id^i}
\]

In this way, the bit at position \( i \) in \( R \) is set to 1 if and only if the \( i \)-th identity matches. The client can extract \( R \) and know the list of matching identities. A few variants of the protocol are also presented in the paper. In the application where the client is only to receive a boolean outcome, like “authenticated/rejected”, the value \( E(R) \) can be computed in this way:

\[
E(R) = E\left(r \cdot \sum_{i=1}^{n} b^i\right) = \prod_{i=1}^{n} E(b^i)^r
\]

where \( r \) is a random integer used to mark the sum of \( b^i \). The client \( C \) is rejected if \( R \) equals 0. In another variant of the protocol, the client also sends to the server an alleged identity \( \hat{id} \) besides its input biometric data. The client is authenticated if and only if its biometric data matches one of the records in the database as well as the alleged identity \( \hat{id} \). The implementation is as follows. After the computation of the encrypted distances \( E(D^i) \), the server also compute:

\[
E(m^i) = E\left(r^i \cdot (\hat{id} - id^i)\right) = (E(\hat{id}) \cdot E(id^i)^{-1})^{r^i}
\]

where \( r^i \) are random integers. Thus all values \( m^i \) will be different from 0 except for the one corresponding to \( \hat{id} \). The values \( E(m^i) \) will be sent to the client during the computation of
If $m^i$ is 0, the client will return the exact $E(b^i)$ of bit $b^i$, otherwise a dummy $E(0)$ is sent. In this way only a single $b^i$ can be not 0 and only if it is corresponding to $\hat{id}$.

In [3], secure protocols for iris code and fingerprint identification are presented. The approaches are analogous to that of Sadeghi [41]. They involve computing Euclidean distances using homomorphic encryption, followed by garbled circuits-based comparisons of the result. The paper also makes some optimizations for the computation.

Huang et al. [15] also propose new methods that substantially reduce the computation and bandwidth costs of each of the phases in typical privacy-preserving biometric matching protocols. In the distance-computation phase, the protocol builds on the Euclidean-distance protocol by Erkin [9] and adopts the packing technique from Sadeghi [41]. The protocol provides an order-of-magnitude improvement in both computation time and bandwidth by using packing more aggressively. In the matching phase, the protocol uses Yao’s garbled-
circuit technique[49]. In the retrieval phase, the protocol uses a new backtracking technique that allows oblivious recovery of the database record corresponding to the closest matching vector. The idea is to use the intermediate wire labels, a by-product of evaluating the garbled circuit in the matching phase, to efficiently perform oblivious retrieval.
5 Secure similar document detection

The secure similar document detection (SSDD) problem was first introduced in [18] and later extended in [27]. In this problem, Alice, who owns a document, wants to detect whether or not Bob’s collection contains a document similar to her document without disclosing Bob’s database to Alice and vice-versa. Jiang [18] proposed two protocols to accomplish this task. They focus on syntactic similarity. A document is represented as a vector of the frequencies of the terms from the document. In this vector space model, similarity can be measured by the cosine of the two normalized vectors. Since cosine can be computed by the dot product of two vectors, computing similarity between two documents is equivalent to computing the dot product of two normalized vectors of the corresponding documents.

Two algorithms are proposed in [18] to calculate dot product in a privacy-preserving way. This first one is the random matrix-based privacy-preserving dot protocol from [48]. The main advantage of this solution is its efficiency. However, this solution is inflexible in achieving better degrees of protection on the input data under different adversary models. So it will not be discussed here, interested readers can look at the original paper. To overcome the drawback, [18] presents an alternative protocol to compute the dot product based on homomorphic encryption. Initially, both parties, Alice and Bob, run a secure protocol to identify the common terms that appear in both datasets. Then each of their documents can be represented as a vector of terms. Each dimension of the vector corresponds to a term in the collection, and each entry contains frequency information of corresponding term. The basic protocol is shown in Figure 4.

The basic protocol is computationally expensive due to a lot of public key operations at both parties. It is mentioned in [18] that the similarity search between two document sets, each containing 500 documents, takes about a week to complete. [27] extends the work in [18] with two optimizations. First, to reduce the number of modular multiplications...
**Input:** Alice’s input is her document vector $\vec{u}$; Bob’s input is his database $D = \{\vec{v}_1, \ldots, \vec{v}_m\}$. Assume $|\vec{u}| = |\vec{v}_i| = n$. Note that the private key is only known to Alice.

**Output:** Alice learns the similarity scores of her document to each document in Bob’s database: $\sigma_1, \ldots, \sigma_m$. Bob learns nothing.

1. Alice:
   (a). Compute $\vec{z}[i] \leftarrow E_{pk}(\vec{u}[i])$ for $i = 1, \ldots, n$;
   (b). Send $\vec{z}$ to Bob;
2. Bob: for $j = 1$ to $m$ do
   (a). $\vec{y}[i] \leftarrow (\vec{z}[i])^{\vec{v}_j[i]}$, for $i = 1, \ldots, n$;
   (b). Compute $E_{pk}(\sigma_j) \leftarrow \prod_{i=1}^{n} \vec{y}[i]$ sends it to Alice
3. Alice: for for $j = 1$ to $m$ do
   (a). Compute $\sigma_j = D_{pk}(E_{pk}(\sigma_j))$

Figure 4: The secure similar document detection protocol

at Bob, they ignore every ciphertext in Alice’s vector where the corresponding plaintext value at Bob is zero. Second, to reduce the number of document comparisons, each party cluster their document collection into $k$ clusters. The idea is to initially compare only the cluster representatives and measure their similarity. If the similarity value is greater than the similarity threshold, the documents in both clusters are compared using the basic protocol in a pairwise manner. In this way, it is not needed to compute the similarity between Alice’s query document and every document in Bob’s collection. The accuracy of this approach is influenced by the clustering algorithm and the similarity threshold. If the clusters generated by the clustering algorithm does not accurately reflect the underlying document distributions, it may result in a significant loss in query precision and recall. The similarity threshold determines the number of similar documents found. If the threshold is set too high, then this approach may not find all similar documents since the representatives of the clusters may not have a perfect similarity score. Alternatively, a low threshold may result in a high number of cluster candidates for pairwise comparison.

Jiang and Samanthula\[19\] propose two SSDD protocols based on the n-gram model. An n-gram is a substring of size $n$ from a given string. If a text document is treated as a one big string(after removing all punctuation marks and whitespaces), under the n-gram model,
each document can be represented by a set of successive n-gram. In general, n-gram model is considered a better document representation than term vector model, because it is language independent, more sensitive to local similarity, simple, and less sensitive to document modifications[19]. Under the n-gram model, the similarity between any two documents can be calculated by set similarity. The paper adopts Jaccard Coefficient (JC) as a measure of set similarity. Suppose \( u \) and \( v \) are two n-gram representations of the corresponding documents, the JC between \( u \) and \( v \) is given by:

\[
JC(u, v) = \frac{|u \cap v|}{|u \cup v|} = \frac{|u \cap v|}{|u| + |v| - |u \cap v|}
\]

In the protocol, the two parties invoke a secure two-party computation protocol to compute \( |u \cap v| \) in an additively split form. Then they run a secure division protocol[7] to compute the JC.

Blundo[4] introduces a protocol for privacy-preserving evaluation of sample set similarity based on Jaccard Coefficient and the private set intersection cardinality (PSI-CA) protocol[6]. To compare two documents, Alice and Bob first create 3-gram sets of their respective documents. Then Alice hashes her 3-grams and computes the exponentiations of the hashes to a random number \( R_a \). Denote this set as \( A \). Alice then sends \( A \) to Bob who, in turn, computes the exponentiations of these values to a random number \( R_b \) and shuffles the set. He also hashes his 3-grams and computes the exponentiations of the hashes to \( R_b \). Denote this set as \( B \). Bob then sends both sets back to Alice. Alice removes \( R_a \) from \( A \) and computes the cardinality of the intersection between \( A \) and \( B \). From this value, she computes the JC.

The limitation of the basic protocol is that its performance depends on the size of the 3-gram set, which is usually very large. To this end, [4] introduce an optimization based on the MinHash technique. A subset of the original 3-gram set is computed using MinHash. According to MinHash theory, the JC of two subsets is a good approximation of the JC of the
original 3-gram sets. Thus the basic protocol is invoked to privately compute the JC between the two subset. The MinHash approximation reduces considerably the computational and communication costs.

Buyrukbailen and Bakiras introduce a solution based on Simhash document fingerprints. Simhash is a dimensionality reduction technique that encodes all the document terms and their frequencies into a fixed-size bit vector. The Simhash fingerprints of two similar documents will only differ in a few bit positions. Thus it reduces the similarity calculation to the Hamming distance computation between two bit vectors. The basic protocol uses homomorphic properties of the ElGamal cryptosystem to securely compute the Hamming distance, which is used as the similarity measure. Furthermore, to enhance the security, proposes the first method to hide the similarity score from Alice. Instead of sending the ciphertexts of scores $\{\sigma_1, \ldots, \sigma_m\}$ to Alice, Bob computes, for each document $D_i$ in his database, the encryptions of $r_j \cdot (\sigma_i - j)$ where $j \in \{0, 1, \ldots, t\}$. Specifically, $r_j$ is a random value that masks the actual similarity score($\sigma_i$) and $t$ is a similarity threshold agreed on by both parties. Then Bob permutes each set of $(t + 1)$ ciphertexts corresponding to document $D_i$, and sends all these ciphertexts back to Alice. The permutations are required in order to prevent Alice form inferring the relation between the ciphertext position and $j$. Alice decrypts the ciphertexts and determines document $D_i$’s similarity score is within the threshold $t$, if and only if one of the $(t + 1)$ values corresponding to $D_i$ is 0.
6 Secure search over multimedia database

6.1 Secure image search

In section 4.1, I have introduced several protocols for secure face recognition, which is a branch of secure image search. Some general secure image search protocols will be reviewed in this section.

Shashank addresses the problem of protecting the privacy of the query image when searching over a public database, where the images in the database are not encrypted. For the database, the server uses an indexing structure, Vocabulary Tree proposed by Nister which has a hierarchical structure. It employs SIFT features which are extracted from the images. They are hierarchically clustered to yield a vocabulary of visual words as well as a vocabulary tree. The images are then indexed using this tree with each leaf node storing a list of image ids that visit it. Given a query image by the user, its feature vector is extracted to query the tree. During querying, one traverses the tree by taking decision at each node to decide which child node to traverse next. This decision is taken by the distance between the query’s feature vector and the data at the node. To prevent the server from knowing which node is accessed by the client, the client uses a PIR solution to fetch the node. Private Information Retrieval (PIR) allows a client to retrieve data at a particular index in a database without revealing the query to the server. When a leaf node is encountered, the data in it is used to get the results.

Avidan presents the problem of oblivious image matching, where two parties want to determine whether they have images of the same object or scene, without revealing any additional information. To this end, they reduce the image matching problem to a two-level version of the fuzzy set matching problem, and then presented a protocol to privately compute this.

[40, 38] also introduce protocols that can keep both the client and the server’s privacy. In
the protocol consists of two parts: secure querying mechanism and oblivious transfer of the decryption keys. The outline of the first part is as follows. The user extracts the feature vector and distorts it with a constant random vector to prevent any statistical inference by the database server. The user then encrypts it and sends it to the server. The server uses the encrypted query feature vector and performs a homomorphic addition with a random feature vector. The server also performs a subtraction of the database image feature vector with the same random feature vector. The server then permute the two subtracted vectors and send them back to the user. The permutation is used so that the user is not able to learn the relative indexing structure of the database images. Upon receiving the data for the server, the user have enough information to compute the Euclidean norm of the numerical difference between the the query feature vector and the database image feature vector. In the oblivious transfer part, the user get a subset of the encrypted images and obliviously get the image decryption keys of his/her choice from the server. Thus the user can decrypt the images selected.

Rane builds a solution using attribute based encryption(ABE). In an ABE system such as [42], Bob obtains some public encryption parameters from a Key Authority and generates his ciphertext that contain two entities: the encryption of the message \( m \) and a so-called attribute vector \( x \). Alice can know the decryption key if and only if her attribute vector \( y \) and and Bob’s \( x \) satisfy a mathematical condition. In the paper, the mathematical condition is based on the Euclidean distance between two attribute vectors. The outline of the protocol is as follows. Let \( M^{(i)}, i \in \{1,2,\ldots,m\} \) represent the images of Bob’s database. For each \( i \), Bob generates a secrete key \( L_i \) for a symmetric cryptosystem of his choice. Using this secret key he generates the ciphertext \( S(M^{(i)}, L_i) \). After this, Bob extracts from each image \( M^{(i)} \), an attribute vector \( x^{(i)} \). The attribute vector is an efficient representation of the image, such as the SIFT feature vector. Then, Bob generates a vector of public encryption parameters \( W \), and computes the ciphertext \( C(x^{(i)}, L_i, W) \). The image retrieval process is
depicted in Figure 5. The essential requirement is that Alice should be able to retrieve an image $M^{(j)}, j \in \{1, 2, \ldots, m\}$ from Bob if and only if her own attribute $y$ satisfies a specific mathematical condition with respect to the image in Bob’s database. Therefore, the ciphertext $C(x^{(j)}, L_j, W)$ is designed such that it can be decrypted by Alice only if a specified function $f(x^{(j)}, y)$ takes a value in a permissible set $\mathcal{A}$. To ensure privacy of both parties, the condition $f(x^{(j)}, y) \in \mathcal{A}$ is checked without revealing the attribute $x^{(j)}$ to Alice and without revealing the attribute $y$ to Bob. Once $L_j$ is decrypted by Alice, she retrieves $S(M^{(j)}, L_j)$ from Bob by oblivious transfer, and retrieves the chosen image. By construction, Alice does not discover any other images in Bob’s database, while OT ensures that Bob does not discover the index of the retrieved image.

Figure 5: The image retrieval process
6.2 Secure audio search

Fanti [10] presents a private audio search protocol based on PIR and feature extraction methods in [14]. Each audio file in the database is represented as a collection of time-dependent, quantized audio features called subfingerprints. The features describe the frequency-domain content over successive, overlapping, and fixed-length time frames. The longer the audio file (such as a song), the more subfingerprints are required to describe it. To enhance performance, the database structure consists of a lookup table and a song directory. The song directory is just a list indexed by song number. The lookup table is a list indexed by subfingerprint. Figure 6 depicts this database setup.

![Figure 6: The server database structure](image)

On the client end, suppose the client’s query is a 3-second audio clip. The client begins by converting the clip into a list of 256 subfingerprints. If the client is trying to find matches for the first subfingerprint 0x0003, the client submits a PIR query to the lookup table for
The server will send the entry at location 0x0003 without knowing it. Now if the client wants all fingerprint blocks that start with 0x0003, it can submit a series of new PIR queries, each of which retrieves a fingerprint block that starts precisely at one of the indices in the list of locations. After obtaining a fingerprint block, the client can compute the Hamming distance between it and his query subfingerprints to decide the similarity score and find the closest match.

In [36], the authors propose two frameworks for privacy-preserving speech recognition. In the first framework, the basic problem (without privacy) is treated as one of statistical pattern classification. Classification is usually performed through a Bayes’ classifier. Let \( C \) be a set of candidate classes to which a recording (audio clip) \( X \) might belong. Let \( P(X|C) \) be the probability distribution of speech recordings \( X \) from class \( C \). \( P(X|C) \) is usually approximated through a parametric model, i.e., \( P(X|C) \approx P(X; \lambda_C) \) where \( \lambda_C \) are the parameters of the class \( C \). Classification is performed as:

\[
\hat{C} = \arg \max_{C \in C} \log P(X; \lambda_C) + \log P(C),
\]

where \( P(C) \) is the a priori bias for class \( C \). [36] then describes the computation of \( \hat{C} \) as a pipeline of primitive computations, and show how to execute each primitive in a computationally secure manner using homomorphic encryptions, secure multiparty computation (SMC), and oblivious transfer. The output of each stage of the pipeline is either distributed across both participants in the form of random additive shares, or arrives at one of the participants in an encrypted form, where the other participant holds the key. Thus, both participants must interact to perform the computations until the final result obtained by the correct participant.

In the second framework, the speech pattern classification task is converted into a string comparison operation [34, 35]. A data-length independent feature vector is firstly derived.
from the speech signal such that string comparison implies a nearest-neighbor classification. Second, these vectors are converted to a collection of bit strings using locality sensitive hashing (LSH) schemes\cite{16}. LSH is inherently not privacy preserving due to its locality sensitive property. To satisfy the privacy constraint, a cryptographic hash function $H[\cdot]$ (such as SHA-256 and MD5) is applied to the LSH results. Classification is performed by counting exact matches. Due to the irreversibility of the hashes, the server is not able to recover the original speech data and privacy is maintained.
7 Conclusion and future work

In this survey, I studied secure processing of similarity queries in various application domains. Processing queries without content leakage is counterintuitive. However, researchers have designed protocols that achieve this goal by using special cryptographic techniques, such as encryption and hashing. I have presented numerous protocols in detail, and briefly showed the proof of correctness and security from some representative papers.

For private location proximity detection, as I discussed in section 3, grid and homomorphic cryptosystems are combined to provide a strong privacy-preserving service. However, since the grid is defined by the protocol, it may not match the user’s area of interest exactly. One of my recent work involves designing a scheme that allows the user to specify his area of interest in arbitrary polygonal shapes. At the heart of my scheme is a two-party protocol for the secure point inclusion problem. It allows the querier to know if the location of the other user is inside the polygon, without letting the user learn the polygon and without letting the querier learn the location of the user. On the other hand, the user may have his own privacy requirement, namely that he does not want to be found within an area smaller than a certain threshold. Unfortunately, this requirement cannot be enforced in our current protocol. One direction that we plan to investigate in the future is to use cryptographic commitment schemes [11] that will allow the querier to commit his input prior to the protocol execution. Then, if the proximity result is true, the querier will be forced to reveal its polygon to the user, in order for him to verify that his privacy was not violated.

For complex data, such as biometric and multimedia data, the secure protocols usually consist of three parts: feature extraction, distance computation, and selection of the matching records. For feature extraction, it is desirable to design algorithms that can generate features that are small in size but also result in the least accuracy loss. A popular research topic is to represent the raw data or features vectors in a way that reduces nearest-neighbor
searches to a Hamming distance comparison, which is computationally very efficient in both the private and non-private domains. Several techniques are used to generate the bit vectors, such as locality-sensitive hashing and simhash. The last two parts of the protocol may be implemented together in some applications. With public image databases (Instagram, Flickr, Facebook, Google Images) being available in recent years, my current research aims to design schemes that can provide secure image search over these large public databases that can maintain the privacy of the query image. For reasons just mentioned, I plan to use bit vectors to represent the images when processing a search. The bit vector of an image is generated from the feature vector (such as SIFT\cite{24}, GIST\cite{31}) of the image. Since the database is large, the images need to be clustered to speed up the search. To sum up, I will try to utilize new techniques or improve the methods that I have surveyed to achieve the best result.
References


