Secure and Efficient Iris and Fingerprint Identification

Marina Blanton
Department of Computer Science and Engineering, University of Notre Dame
mblanton@nd.edu

Paolo Gasti
Department of Computer Science, New York Institute of Technology
pgasti@nyit.edu

Abstract - Advances in biometric recognition and the increasing use of biometric data prompt significant privacy challenges associated with the possible misuse, loss, or theft of biometric data. Biometric comparisons are often performed by two mutually distrustful parties, one of which holds a biometric sample while the other owns a possibly large collection of biometric data. Due to privacy and liability considerations, neither party is willing to share its data. This gives rise to the need to utilize secure computation techniques over biometric data where no information is revealed to the parties except the outcome of the comparison or search for identification purposes. In this chapter, we present the design, security analysis, and performance of privacy-preserving identification protocols for iris codes and fingerprints. Combined with certain optimizations, such techniques are suitable for practical use on large data sets.

Index Terms - Privacy-Preserving Protocols, Biometric Identification, Secure Multiparty Computation, Iris, Fingerprints

1 Introduction

Recent advances in biometric recognition have made the use of biometric information more prevalent for verification and identification purposes. Large-scale collections of biometric data in use today include, for example, fingerprint, face, and iris images collected by the US Department of Homeland Security (DHS) from visitors [48]; fingerprint and iris images collected by the government of India from (more than billion) citizens [56]; iris, fingerprint, and face images collected by the United Arab Emirates (UAE) Ministry of Interior from visitors [57]; and adoption of biometric passports in several countries. While biometric systems serve as an excellent tool for authentication and identification of individuals, biometric data is undeniably extremely sensitive and must 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 agencies. However, there could be legitimate reasons to carry out computations on biometric data belonging to different entities. For example, a non-government agency may need to know whether a biometric sample it possesses belongs to an individual on the government watch-list. In this case the agency would like to maintain the privacy of the individual if no matches are found, and the government also does not want to release its database to third parties.

The above requires carrying out computation over biometric data in a way that keeps the data private and reveals only the outcome of the computation. In particular, we treat the problem of *biometric identification*, where a client C is in a possession of a biometric sample X and a server S possesses a biometric

database D. The client would like to know whether there is any X' in D matching X by comparing X to each biometric record in D. The computation amounts to comparing X to each $Y \in D$ in a privacy-preserving manner. This formulation is general enough to apply to a number of other scenarios, ranging from a single comparison of X and Y to the case where two parties need to compute the set of biometric data records common to their respective databases. We assume that the result of comparing X and Y is a bit, and no additional information about X or Y should be learned by the parties as a result of secure computation. Throughout this chapter, we also assume that X and Y correspond to biometric templates, i.e., they have representations suitable for biometric comparison after raw biometric samples have been processed by a feature extraction algorithm. Feature extraction can be performed for each biometric sample independently, and we do not discuss this further.

This chapter introduces and discusses secure techniques that perform the aforementioned computation with provable protection of data privacy. We present protocols for two types of biometric characteristics: iris and fingerprints. While iris codes are normally represented as binary strings and use very similar matching algorithms, there is a variety of representations and comparison algorithms for fingerprints. For that reason, we study two types of matching algorithms for fingerprints: (i) FingerCodes that use fixed-size representations and a simple comparison algorithm and (ii) a traditional and most widely used method for pairing minutia points in one fingerprint with minutiae in another fingerprint. With such techniques, the outcome of the computation can be made available to either party or both of them; for concreteness, in our description we have the client learn the outcome of each comparison.

2 Description of Computation

Without loss of generality, in what follows we assume that client C holds a single biometric template X and server S holds a database of biometric data D. The goal is to learn whether C's biometric template has a match in S's database without learning any additional information. This is accomplished by comparing X to each biometric template $Y \in D$, and as a result of each comparison C learns a bit that indicates whether the comparison resulted in a match.

2.1 Iris

Let an iris code X be represented as an m-bit binary string. We use X_i to denote the i-th bit of X. In iris-based recognition, after feature extraction, biometric comparison is normally performed by computing the *normalized* Hamming distance between two biometric representations. (To simplify presentation, we refer to *normalized Hamming distance* simply as *Hamming distance* henceforth.) Furthermore, the feature extraction process is such that some bits of the extracted string X are unreliable and are ignored in the comparison process. Information about such bits is stored in an additional m-bit string, called *mask*, where its i-th bit is set to 1 if the i-th bit of X should be used in the comparison process and is set to 0 otherwise. For iris code X, we use $M(X)$ to denote the mask associated with X. Often, a predetermined number of bits (e.g., 25% in [31] and 35% in [6]) are considered unreliable in each biometric template. Thus, to compare two biometric templates X and Y, their Hamming distance takes into account the respective masks. That is, if the Hamming distance between two iris codes without masks is computed as:

$$HD(X,Y) = \frac{||X \oplus Y||}{m} = \frac{\sum_{i=1}^{m}(X_i \oplus Y_i)}{m}$$

the computation of the Hamming distance that uses masks becomes [23]:

$$HD(X,M(X),Y,M(Y)) = \frac{||(X \oplus Y) \cap M(X) \cap M(Y)||}{||M(X) \cap M(Y)||} \quad (1)$$

In other words, we have

$$HD(X, M(X), Y, M(Y)) = \frac{\sum_{i=1}^{m}((X_i \oplus Y_i) \wedge M(X_i) \wedge M(Y_i))}{\sum_{i=1}^{m}(M(X_i) \wedge M(Y_i))}.$$

Throughout this chapter, we assume that the latter formula is used and simplify the notation to $HD(X, Y)$. Then the computed Hamming distance is compared with a specific threshold T, and the biometric samples X and Y are considered to be a match if the distance is below the threshold, and a mismatch otherwise. The threshold T is chosen based on the distributions of authentic and impostor data. (In the likely case of overlap of the two distributions, the threshold is set to achieve the desired levels of false accept and false reject rates based on the security goals.)

Two iris representations can be slightly misaligned. This problem is usually caused by head tilt during image acquisition. To account for this, the matching process attempts to compensate for the error and rotates a biometric representation by a fixed amount to determine the lowest distance. More precisely, each iris code is represented as a two-dimensional bit array and rotation corresponds to a circular shift which is applied to each row. Each biometric is then rotated to the left and to the right a small fixed number of times, which we denote by c. The minimum Hamming distance across all rotations is then compared to the threshold. That is, if we let $LS^j(\cdot)$ (resp., $RS^j(\cdot)$) denote a circular left (resp., right) shift of the argument by a fixed number of bits (normally 2 bits due to the properties of the feature extraction process), the matching process becomes:

$$\min(HD(X, LS^c(Y)), \ldots, HD(X, LS^1(Y)), HD(X, Y), HD(X, RS^1(Y)), \ldots, HD(X, RS^c(Y))) \overset{?}{<} T \qquad (2)$$

Throughout this chapter, we assume that the algorithms for comparing two biometric samples are public, as well as any constant thresholds T. The protocols we present, however, maintain their security and computational performance guarantees even if the (fixed) thresholds are known only to the server who owns the database.

2.2 Fingerprints

Work on fingerprint identification dates back to the late 1800s, with a number of different approaches currently available (see, e.g., [46] for an overview). The most popular and widely used techniques extract information about minutiae from a fingerprint and store that information as a set of points in the two-dimensional plane. Fingerprint comparison in this case consists of finding a matching between two sets of points so that the number of paired minutiae is maximized. In more detail, a biometric template X is represented as a set of m_X points $X = \langle (x_1, y_1, \alpha_1), \ldots, (x_{m_X}, y_{m_X}, \alpha_{m_X}) \rangle$, where x_i and y_i are coordinates of minutia i in a two-dimensional space and α_i is its orientation (represented as an angle in degrees). A minutia $X_i = (x_i, y_i, \alpha_i)$ in X and minutia $Y_j = (x'_j, y'_j, \alpha'_j)$ in Y are considered matching if the spatial (i.e., Euclidean) distance between them is smaller than a given threshold d_0 and the directional difference between them is smaller than a given threshold α_0. That is, we compute this as:

$$\sqrt{(x'_j - x_i)^2 + (y'_j - y_i)^2} < d_0 \quad \text{and} \quad \min(|\alpha'_j - \alpha_i|, 360^\circ - |\alpha'_j - \alpha_i|) < \alpha_0. \qquad (3)$$

It is necessary to tolerate small differences in the position and orientation of minutia points to account for errors introduced by feature extraction algorithms (e.g., quantization) and small skin distortions. Two points within a single fingerprint are also assumed to lie within at least distance d_0 of each other.

Before two fingerprints can be compared, they need to be pre-aligned, which maximizes the number of matching minutiae. We can distinguish two types of alignment: absolute and relative. With absolute alignment, each fingerprint is pre-aligned independently using the core point or other information. With

relative alignment, information contained in two biometric samples is used to guide their alignment relative to each other. While relative pre-alignment can be more accurate that absolute pre-alignment, such techniques are not practical within a privacy-preserving protocol due to the additional overhead, and we assume that absolute pre-alignment is used. To increase the accuracy of the matching process, a single fingerprint can be stored using a small number of representations with slightly different alignments, and the result of the comparison is a match if at least one of them matches the biometric template being queried.

A simple way used for determining a pairing between minutiae of fingerprints X and Y consists of pairing a minutia X_i with the closest minutia Y_j in Y. Let $mm(X_i, Y_j)$ denote the minutiae matching predicate in Equation 3. Then the pairing function $P(\cdot)$ that determines the mapping of minutiae in X and Y can be defined as follows: for $i = 1, \ldots, m_X$, $P(i) = j$ if Y_j is the closest to X_i among all $Y_k \in Y$ such that $mm(X_i, Y_k) = 1$, and $P(i) =$? if no such Y_j exists. Because each minutia Y_j can be paired with at most one minutia from X, the above algorithm needs to mark all minutiae in Y that have already been paired with a point in X to enforce this constraint.

The above approach will not find the optimum assignment (i.e., the one that maximizes the number of mates) because to find such a pairing a minutia X_i might need to be paired with another minutia Y_j which is not the closest to X_i. The optimum pairing can be achieved by formulating the problem as an instance of minimum-cost maximum flow, where fingerprints X and Y are used to create a flow network. Then this problem can be solved using one of the known algorithms such as Ford-Fulkerson [26] and others. In particular, [37, 58] use a flow network representation of minutia pairing problem to find an optimal pairing, where there is an edge from a node corresponding to minutia $X_i \in X$ to $Y_j \in Y$ iff $mm(X_i, Y_j) = 1$. We refer the reader to [37, 58] for additional detail. For fingerprints consisting of m minutiae, the optimal pairing can be found in $O(m^2)$ time using Ford-Fulkerson algorithm because each minutia from X is connected to at most a constant number of minutiae from Y. In a privacy-preserving setting, however, when information about connections between minutiae in X and Y (and thus the structure of the graph) must remain private, the complexity of this approach would increase. For example, a solution based on [11] would result in complexity $O(m^3 \log m)$, which is substantially slower even for modest values of m. We therefore implement the pairing approach based on the minimum distance outlined above. The algorithm is not guaranteed to find the optimal pairing, but performs well in the privacy-preserving setting.

For the purposes of this chapter, we assume that during fingerprint identification the number of minutiae in a pairing is compared to a fixed threshold T. If in specific fingerprint comparison algorithms this threshold is not constant, but rather is a function of biometric templates X and Y being compared (e.g., a function of the number of points in each template), our techniques can be easily extended to accommodate those variations as well.

Fingerprint matching can also be performed using a different type of information extracted from a fingerprint image. One example is FingerCode [36] which uses texture information from a fingerprint scan to form fingerprint representation X. While FingerCodes are not as distinctive as minutia-based representations and are best suited for use in combination with minutiae to improve the overall accuracy of fingerprint comparisons [46], FingerCode-based identification can be implemented very efficiently in a privacy-preserving protocol. In particular, each FingerCode consists of a fixed number m elements of ℓ bits each. Then FingerCodes $X = (x_1, \ldots, x_m)$ and $Y = (y_1, \ldots, y_m)$ are considered a match if the Euclidean distance between their elements is below the threshold T:

$$\sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \stackrel{?}{<} T \tag{4}$$

The technique we present in this chapter is substantially faster, in terms of both computation and communication, than earlier techniques (e.g., from [5]). We then also proceed with providing a secure protocol for more accurate (but less efficient) minutia-based matching algorithm.

3 Cryptographic Preliminaries

Security model. Intuitively, the level of security that a privacy-preserving construction should achieve is the same as having the participants privately send their inputs to a trusted third party who performs the computation and privately sends the result back. Then a secure technique should provide the same level of data privacy, but without assuming the existence of such a trusted third party.

Our security model is the standard model for secure two-party computation in presence of semi-honest participants [27] (also known as honest-but-curious or passive). In particular, it means that the parties follow the prescribed behavior, but might try to compute additional information from the information obtained during protocol execution. Security in this setting is defined using simulation argument: the protocol is secure if the view of protocol execution for each party is computationally indistinguishable from the view simulated using that party's input and output only. This means that the protocol execution does not reveal any additional information to the participants. The definition below formalizes the notion of security for two semi-honest participants:

Definition 1 *Let parties P_1 and P_2 engage in a protocol π that computes function $f(in_1, in_2) = (out_1, out_2)$, where in_i and out_i denote input and output of party P_i, respectively. Let $\text{VIEW}_\pi(P_i)$ denote the view of participant P_i during the execution of protocol π. More precisely, P_i's view is formed by its input, internal random coin tosses r_i, and messages $m_1, \ldots, m_t$ passed between the parties during protocol execution:*

$$\text{VIEW}_\pi(P_i) = (in_i, r_i, m_1, \ldots, m_t).$$

We say that protocol π is secure against semi-honest adversaries if for each party P_i there exists a probabilistic polynomial time simulator S_i such that

$$\{S_i(in_i, f_i(in_1, in_2)), f(in_1, in_2)\} \equiv \{\text{VIEW}_\pi(P_i), (out_1, out_2)\},$$

where $f_i(in_1, in_2)$ denotes the ith element that $f(in_1, in_2)$ outputs and "$\equiv$" denotes computational indistinguishability.

Homomorphic encryption. Our constructions use a semantically secure additively homomorphic encryption scheme. Informally, semantic security means that a computationally bounded adversary cannot learn any information about the encrypted message from the ciphertext with more than negligible probability in the security parameter (see, e.g., [27] for a formal definition). In an additively homomorphic encryption scheme defined by three algorithms $(Setup, Enc, Dec)$, $Enc(m_1) \cdot Enc(m_2) = Enc(m_1 + m_2)$ for any two plaintexts m_1 and m_2, which also implies that $Enc(m_1)^{m_2} = Enc(m_2 \cdot m_1)$, where plaintext m_2 is known. While any encryption scheme with the above properties (such as the well known Paillier encryption scheme [50]) suffices for the purposes of this chapter, the construction due to Damgård et al. [21, 20] (DGK) is of particular interest here.

To be able to understand optimizations used in our techniques, we briefly describe the relevant encryption schemes. In Paillier encryption scheme, a public key consists of a k-bit RSA modulus $N = pq$, where p and q are prime numbers of bitlength $k/2$ and k is the security parameter, and an element g whose order is a multiple of N in $\mathbb{Z}^*_{N^2}$. Given a message $m \in \mathbb{Z}_N$, encryption is performed as $Enc(m) = g^m r^n \bmod N^2$, where $r \overset{R}{\leftarrow} \mathbb{Z}_N$ and notation $a \overset{R}{\leftarrow} A$ means that a is chosen uniformly at random from the set A. In DGK encryption scheme [21, 20], which was designed to work with small plaintext spaces and has shorter ciphertext size than other randomized encryption schemes, a public key consists of (i) a (small, possibly prime) integer u that defines the plaintext space, (ii) k-bit RSA modulus $N = pq$ such that p and q are $k/2$-bit primes, v_p and v_q are t-bit primes for another security parameter t (smaller than k), and $uv_p|(p-1)$ and $uv_q|(q-1)$, and (iii) elements $g, h \in \mathbb{Z}^*_N$ such that g has order uv_pv_q and h has order v_pv_q. Given a

message $m \in \mathbb{Z}_u$, encryption is performed as $Enc(m) = g^m h^r \bmod N$, where $r \overset{R}{\leftarrow} \{0,1\}^{2.5t}$. We refer the reader to the original publications [50] and [21, 20], respectively, for any additional information.

Garbled circuit evaluation. Originated in Yao's work [59], garbled circuit evaluation allows two parties to securely evaluate any function represented as a boolean circuit. The basic idea is that, given a Boolean circuit composed of gates, one party P_1 creates a garbled circuit by assigning to each wire two randomly chosen labels (one corresponding to 0 and the other corresponding to 1). P_1 also encodes gate information in a way that given labels corresponding to the input wires (encoding specific inputs), the label corresponding to the output of the gate on those inputs can be recovered. The second party, P_2, evaluates the circuit using labels corresponding to inputs of both P_1 and P_2 (without learning anything in the process since P_2 does not know the meaning of the (random) labels that it sees during evaluation). At the end, the result of the computation can be recovered by linking the computed output labels to the bits which they encode.

Recent literature provides optimizations that reduce computation and communication overhead associated with circuit construction and evaluation. Kolesnikov and Schneider [40] describe an optimization that permits XOR gates to be evaluated for free, i.e., there is no communication overhead associated with such gates and their evaluation does no involve cryptographic functions. Pinkas et al. [51] additionally give a mechanism for reducing communication complexity of binary gates by 25%: now each gate can be specified by encoding only three outcomes of the gate instead of all four. Finally, Kolesnikov et al. [39] improve the complexity of certain commonly used operations such as addition, multiplication, comparison, etc. by reducing the number of non-XOR gates: adding two n-bit integers requires $5n$ gates, n of which are non-XOR gates; comparing two n-bit integers requires $4n$ gates, n of which are non-XOR gates; and computing the minimum of t n-bit integers (without the location of the minimum value) requires $7n(t-1)$ gates, $2n(t-1)$ of which are non-XOR gates. Garbling and evaluation of large circuits can also be pipelined [32], so that the entire circuit does not have to reside in memory.

With the above techniques, evaluating a non-XOR gates involves one invocation of the hash function [40] (which is assumed to be correlation robust [42]) or one call to AES [7]. During garbled circuit evaluation, P_2 directly obtains keys corresponding to P_1's inputs from P_1 and engages in the oblivious transfer (OT) protocol to obtain keys corresponding to P_2's inputs.

Oblivious Transfer. In 1-out-of-2 Oblivious Transfer, OT_1^2, one party, the sender, has as its input two strings 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 strings held by the sender. There is a rich body of research literature on OT, and in this chapter we use its efficient implementation from [47] as well as OT extension from [35] that reduces a large number of OT protocol executions to κ of them, where κ is the security parameter. This, in particular, means that obtaining the keys corresponding to P_2's inputs in garbled circuit evaluation by P_2 incurs only small overhead. We note that there are other very recent OT extensions such as [38] and [2] that further reduce the cost of OT and their usage in our solution will reduce the overhead that we report.

4 Secure Iris Identification

In this section, we present our solution for biometric identification based on iris codes. Our solution combines homomorphic encryption with garbled circuit evaluation. The rationale behind building hybrid protocols is that the use of homomorphic encryption will allow an encrypted template X to be used in comparisons to many $Y \in D$, while garbled circuits are very fast for certain operations such as comparisons, where techniques based on homomorphic encryption are much more costly.

4.1 Structural Optimization of the Computation

As indicated in Equation 1, computing the distance between two iris codes involves securely evaluating the division operation. While techniques for carrying out this operation using secure multi-party computation are known (see, e.g., [3, 17, 9, 18]), their performance in practice is often substantially slower than performance of other elementary operations, which poses a problem for this application. For an example, according to [8], two-party evaluation of garbled circuits for division produced by Fairplay [45] takes several seconds for numbers of length 24--28 bits, but circuits for longer integers could not be constructed due to the rapidly increasing memory requirements of Fairplay. More recent results achieve faster circuit evaluation, but performance of this operation normally is not reported.[1] Fortunately, in our case the computation can be rewritten to completely avoid this operation and replace it with multiplication. That is, using the notation

$$HD(X,Y) = ||(X \oplus Y) \cap M(X) \cap M(Y)|| \,/\, ||M(X) \cap M(Y)|| = D(X,Y) \,/\, M(X,Y),$$

instead of testing whether $HD(X,Y) \stackrel{?}{<} T$, we can test whether $D(X,Y) \stackrel{?}{<} T \cdot M(X,Y)$. While the computation of the minimum distance as used in Equation 2 is no longer possible, we can replace it with equivalent computation that does not increase its cost. Now the computation becomes:

$$D(X, LS^c(Y)) \stackrel{?}{<} T \cdot M(X, LS^c(Y)) \vee \cdots \vee D(X, RS^c(Y)) \stackrel{?}{<} T \cdot M(X, RS^c(Y)) \tag{5}$$

When this computation is carried out over real numbers, T lies in the range [0, 1]. In our case, it is desirable to carry out the computation over the integers, which means that we ''''scale up'' all values with the desired level of precision. That is, by using ℓ bits to achieve desired precision, we multiply $D(X,Y)$ by 2^ℓ and let T range between 0 and 2^ℓ. Now $2^\ell D(X,Y)$ and $T \cdot M(X,Y)$ can be represented using $\lceil \log m \rceil + \ell$ bits.

4.2 Base Protocol

In what follows, we first describe the basic privacy-preserving protocol for iris codes. The consecutive section presents optimizations and the resulting performance of the protocol. At the high level, the solution consists of using encrypted data to compute (partial) distances between biometric templates, after which we switch to garbled circuit evaluation for finishing the computation and producing the final result.

With this approach, the client C generates a public-private key pair (pk, sk) for an additively homomorphic encryption scheme and distributes the public key pk. This is a one-time setup cost for the client for all possible invocations of this protocol with any number of servers. During the protocol itself, the secure computation proceeds as specified in Equation 5. In the beginning, C sends its inputs encrypted with pk to the server S. At the server side, the computation first proceeds using homomorphic encryption, but later the client and the server convert the intermediate result into a secret-shared form and finish the computation using garbled circuit evaluation. As mentioned before, we employ this structure due to the fact that secure two-party computation of the comparison operation is significantly faster using garbled circuit evaluation, but the rest of the computation in our case is best performed on encrypted values.

To compute $D(X,Y) = \sum_{i=1}^{m} (X_i \oplus Y_i) \wedge M(X_i) \wedge M(Y_i)$ using algebraic computation, we use $X_i \oplus Y_i = X_i(1 - Y_i) + (1 - X_i)Y_i$ and obtain:

$$D(X,Y) = \sum_{i=1}^{m} (X_i(1 - Y_i) + (1 - X_i)Y_i)M(X_i)M(Y_i).$$

[1] Secure evaluation of the division operation in the multi-party setting was reported in [30, 12, 1], but such techniques cannot be directly used in the two-party setting that we employ.

Input: C has biometric template X, $M(X)$ and key pair (pk, sk); S has a database D composed of biometric templates in the form Y, $M(Y)$.
Output: C learns what records in D resulted in match with X if any, i.e., it learns a bit as a result of comparison of X with each $Y \in D$.
Protocol steps:

1. For each $i = 1, \ldots, m$, C computes encryptions $\langle a_{i1}, a_{i2} \rangle = \langle Enc(X_i M(X_i)), Enc((1 - X_i)M(X_i)) \rangle$ and sends them to S.
2. For each $i = 1, \ldots, m$, S computes encryption of $M(X_i)$ by setting $a_{i3} = a_{i1} \cdot a_{i2} = Enc(X_i M(X_i)) \cdot Enc((1 - X_i)M(X_i)) = Enc(M(X_i))$.
3. For each record Y in the database, S and C perform the following steps in parallel:
   (a) For each amount of shift $j = -c, \ldots, 0, \ldots, c$, S rotates the bits of Y by the appropriate number of positions to obtain Y^j and proceeds with all Y^j's in parallel.
      i. To compute $(X_i \oplus Y_i^j)M(X_i)M(Y_i^j) = (X_i(1-Y_i^j)+(1-X_i)Y_i^j)M(X_i)M(Y_i^j)$ in encrypted form, S computes $b_i^j = a_{i1}^{(1-Y_i^j)M(Y_i^j)} \cdot a_{i2}^{Y_i^j M(Y_i^j)} = Enc(X_i M(X_i)(1 - Y_i^j)M(Y_i^j) + (1 - X_i)M(X_i)Y_i^j M(Y_i^j))$.
      ii. S adds the values contained in b_i^j's to obtain $b^j = \prod_{i=1}^m b_i^j = Enc(\sum_{i=1}^m (X_i \oplus Y_i^j)M(X_i)M(Y_i^j)) = Enc(||(X \oplus Y^j) \cap M(X) \cap M(Y^j)||)$. S then "lifts up" the result, blinds, and randomizes it as $c^j = (b^j)^{2^\ell} \cdot Enc(r_S^j)$, where $r_S^j \xleftarrow{R} \{0,1\}^{\lceil \log m \rceil + \ell + \kappa}$, and sends the resulting c^j to C.
      iii. To obtain $T(||M(X) \cap M(Y^j)||)$, S computes $d_i^j = a_{i3}^{M(Y_i^j)} = Enc(M(X_i) \cdot M(Y_i^j))$ and $d^j = (\prod_{i=1}^m d_i^j)^T = Enc(T(\sum_{i=1}^m M(X_i)M(Y_i^j)))$. S blinds and randomizes the result as $e^j = d^j \cdot Enc(t_S^j)$, where $t_S^j \xleftarrow{R} \{0,1\}^{\lceil \log m \rceil + \ell + \kappa}$, and sends e^j to C.
      iv. C decrypts the received values and sets $r_C^j = Dec(c^j)$ and $t_C^j = Dec(e^j)$.
   (b) C and S perform $2c + 1$ comparisons and OR of the results of the comparisons using garbled circuit. C enters r_C^j's and t_C^j's, S enters $-r_S^j$'s and $-t_S^j$'s, and C learns bit b computed as $\bigvee_{j=-c}^{c}((r_C^j - r_S^j) \overset{?}{<} (t_C^j - t_S^j))$. To achieve this, S creates the garbled circuit and sends it to C. C obtains keys corresponding to its inputs using OT, evaluates the circuit, and S sends to C the key-value mapping for the output gate.

Figure 1: Secure two-party protocol for iris identification.

$M(X, Y)$ is computed as $\sum_{i=1}^m M(X_i)M(Y_i)$. Then if the server obtains encryptions of $X_i M(X_i)$, $(1-X_i)M(X_i)$, and $M(X_i)$ for each i from the client, it will be able to compute $D(X, Y)$ and $M(X, Y)$ using its knowledge of the Y_i's and the homomorphic properties of the encryption. Figure 1 describes the protocol, in which after receiving C's encrypted values S produces $Enc(M(X_i))$'s and proceeds to compute $D(X, Y^j)$ and $M(X, Y^j)$ in parallel for each Y in its database. Here Y^j denotes biometric template Y shifted by j positions and j ranges from $-c$ to c. At the end of steps 3(a).i and 3(a).ii the server obtains $Enc(2^\ell D(X, Y^j) + r_S^j)$ for a randomly chosen r_S^j of its choice, and at the end of step 3(a).iii S obtains $Enc(T \cdot M(X, Y^j) + t_S^j)$ for a random t_S^j of its choice. The server sends these values to the client who decrypts them. Therefore, at the end of step 3(a) C holds $r_C^j = 2^\ell D(X, Y^j) + r_S^j$ and $t_C^j = T \cdot M(X, Y^j) + t_S^j$ and S holds $-r_S^j$ and $-t_C^j$, i.e., they additively share $2^\ell D(X, Y^j)$ and $T \cdot M(X, Y^j)$.

What remains to compute is $2c + 1$ comparisons (one per each Y^j) followed by $2c$ OR operations as specified by Equation 5. This is accomplished using garbled circuit evaluation, where C enters r_C^j's and t_C^j's and S enters r_S^j's and t_S^j's and they learn a bit, which indicates whether Y was a match.

Note that since r_C^j's, r_S^j's, t_C^j's and t_S^j's are used as inputs to the garbled circuit and will need to be added inside the circuit, we want them to be as small as possible. Therefore, instead of providing unconditional hiding by choosing t_S^j and r_C^j from $\mathbb{Z}_N^*$ (where N is from pk), the protocol achieves statistical hiding by choosing these random values to be κ bits longer than the values that they protect, where κ is a security parameter (so that the value t^j revealed to C statistically hides the computed distance).

4.3 Optimizations

Pre-computation and offline communication. Similar to other literature on secure biometric identification, we distinguish between offline and online stages, where any computation and communication that does not depend on the inputs of the participating parties can be moved to the offline stage. In our protocol, first notice that most modular exponentiations (the most expensive operation in the encryption scheme) can be precomputed. That is, the client needs to produce $2m$ encryptions of bits. Because both m and the average number of 0's and 1's in a biometric template and a mask are known, the client can produce a sufficient number of bit encryptions in advance. In particular, X normally will have 50% of 0's and 50% of 1's, while 75% (or a similar number) of $M(X)$'s bits are set to 1 and 25% to 0 during processing. Let p_0 and p_1 (q_0 and q_1) denote the fraction of 0's and 1's in an iris code (resp., its mask), where $p_0 + p_1 = q_0 + q_1 = 1$. Therefore, to have a sufficient supply of ciphertexts to form tuples $\langle a_{i1}, a_{i2}\rangle$, the client needs to precompute $(2q_0 + q_1(p_1 + \varepsilon) + q_1(p_0 + \varepsilon))m = (1 + q_0 + 2q_1\varepsilon)m$ encryptions of 0 and $(q_1(p_1 + \varepsilon) + q_1(p_0 + \varepsilon))m = q_1(1 + 2\varepsilon)m$ encryptions of 1, where ε is used as a cushion since the number of 0's and 1's in X might not be exactly p_0 and p_1, respectively. Then at the time of the protocol the client simply uses the appropriate ciphertexts to form its transmission.

Similarly, the server can precompute a sufficient supply of encryptions of r_S^j's and t_S^j's for all records. That is, the server needs for produce $2(2c+1)|D|$ encryptions of different random values of length $\lceil \log m \rceil + \ell + \kappa$, where $|D|$ denotes the size of the database D. The server also generates one garbled circuit per record Y in its database (for step 3(b) of the protocol) and communicates the circuits to the client. In addition, the most expensive part of the oblivious transfer can also be performed during the offline stage, as detailed below.

Optimized multiplication. Server's computation in steps 3(a).i and 3(a).iii of the protocol can be significantly lowered as follows. To compute ciphertexts b_i^j, S needs to calculate $a_{i1}^{(1-Y_i^j)M(Y_i^j)} \cdot a_{i2}^{Y_i^j M(Y_i^j)}$. Since the bits Y_i^j and $M(Y_i^j)$ are known to S, this computation can be rewritten using one of the following cases:

- $Y_i^j = 0$ and $M(Y_i^j) = 0$: in this case both $(1 - Y_i^j)M(Y_i^j)$ and $Y_i^j M(Y_i^j)$ are zero, which means that b_i^j should correspond to an encryption of 0 regardless of a_{i1} and a_{i2}. Instead of having S create an encryption 0, we set b_i^j to the empty value, i.e., it is not used in the computation of b^j in step 3(a).ii.
- $Y_i^j = 1$ and $M(Y_i^j) = 0$: the same as above.
- $Y_i^j = 0$ and $M(Y_i^j) = 1$: in this case $(1 - Y_i^j)M(Y_i^j) = 1$ and $Y_i^j M(Y_i^j) = 0$, which means that S sets $b_i^j = a_{i1}$.
- $Y_i^j = 1$ and $M(Y_i^j) = 1$: in this case $(1 - Y_i^j)M(Y_i^j) = 0$ and $Y_i^j M(Y_i^j) = 1$, and S therefore sets $b_i^j = a_{i2}$.

The above implies that only $q_1 m$ ciphertexts b_i^j need to be added in step 3(a).ii to form b^j (i.e., $q_1 m - 1$ modular multiplications to compute the hamming distance between m-element strings).

Similar optimization applies to the computation of d_i^j and d^j in step 3(a).iii of the protocol. That is, when $M(Y_i^j) = 0$, d_i^j is set to the empty value and is not used in the computation of d^j; when $M(Y_i^j) = 1$, S sets $d_i^j = a_{i3}$. Consequently, $q_1 m$ ciphertexts are used in computing d^j.

To further reduce the number of modular multiplications, we can adopt the idea from [49], which consists of precomputing all possible combinations for ciphertexts at positions i and $i + 1$ and reducing the number of modular multiplications used during processing a database record in half. In our case, the value of $b_i^j b_{i+1}^j$ requires computation only when $M(Y_i^j) = M(Y_{i+1}^j) = 1$. In this case, computing $a_{i1}a_{(i+1)1}$, $a_{i1}a_{(i+1)2}$, $a_{i2}a_{(i+1)1}$, and $a_{i2}a_{(i+1)2}$, for each odd i between 1 and $m - 1$ will cover all possibilities.

Note that these values need to be computed once for all possible shift amounts of the biometric data (since only server's Y's are shifted). Depending on the distribution of the set bits in each $M(Y)$, the number of modular multiplication now will be between $q_1 m/2$ (when $M(Y_i) = M(Y_{i+1})$ for each odd i) and $m(q_0 + (1-2q_0)/2) = m/2$ (when $M(Y_i) \neq M(Y_{i+1})$ for as many odd i's as possible). This approach can be also applied to the computation of d^j (where only the value of $a_{i3}a_{(i+1)3}$ needs to be precomputed for each odd i) resulting in the same computational savings during computation of the hamming distance. Furthermore, by precomputing the combinations of more than two values additional savings can be achieved during processing of each Y.

Optimized encryption scheme. As it is clear from the protocol description, its performance crucially relies on the performance of the underlying homomorphic encryption scheme for encryption, addition of two encrypted values, and decryption. Instead of utilizing a general purpose encryption scheme such as Paillier, we turn our attention to schemes of restricted functionality which promise to offer improved efficiency. In particular, the DGK additively homomorphic encryption scheme [21, 20] was developed to be used for secure comparison, where each ciphertext encrypts a bit. In that setting, it has faster encryption and decryption time than Paillier and each ciphertext has size k using a k-bit RSA modulus (while Paillier ciphertext has size $2k$). To be suitable for our application, the DGK scheme can be modified to work with plaintexts longer than one bit used in its original design. In that case, at decryption time, one needs to additionally solve the discrete logarithm problem where the base is 2-smooth using Pohlig-Hellman algorithm. This means that decryption uses additional $O(n)$ modular multiplications for n-bit plaintexts. Now recall that in the protocol we encrypt messages of length $\lceil \log m \rceil + \ell + \kappa$ bits. The use of the security parameter κ significantly increases the length of the plaintexts. We, however, notice that the DGK encryption can be setup to permit arithmetic on encrypted values such that all computations on the underlying plaintexts are carried modulo 2^n for any n. For our protocol it implies that (i) the blinding values r_S^j and t_S^j can now be chosen from the range $[0, 2^n - 1]$, where $n = \lceil \log m \rceil + \ell$, and (ii) this provides information-theoretic hiding (thus improving the security properties of the protocol). This observation has a profound impact not only on the client decryption time in step 3(a).iv (which decreases by about an order of magnitude), but also on the consecutive garbled circuit evaluation, where likewise the circuit size is significantly reduced in size.

Circuit construction. We construct garbled circuits using the most efficient techniques from [51] and references therein. By performing addition modulo 2^n and eliminating gates which have a constant value as one of their inputs, we reduce the complexity of the circuit for addition to $n-1$ non-XOR gates and $5(n-1)-1$ total gates. Similarly, after eliminating gates with one constant input, the complexity of the circuit for comparison of n-bit values becomes n non-XOR gates and $4n-2$ gates overall. Since in the protocol there are two additions and one comparison per each j followed by $2c$ OR gates, the size of the overall circuit is $14(n-1)(2c+1)+2c$ gates, $(3n-2)(2c+1)+2c$ of which are non-XOR gates. Note that this circuit does not use multiplexers, which are required (and add complexity) during direct computation of minimum.

Oblivious transfer. The above circuit requires each party to supply $2n(2c+1)$ input bits, and a new circuit is used for each Y in D. Similar to some other techniques, the combination of fast OT and OT extension (we use [35] and [47]) achieves the best performance in our case. Let the server create each circuit and the client evaluate them. Using the results of [35], performing OT_1^2 the total of $2n(2c+1)|D|$ times, where the client receives a κ-bit string as a result of each OT for a a security parameter κ, can be reduced to κ invocations of OT_1^2 (that communicates to the receiver κ-bit strings) at the cost of $4\kappa \cdot 2n(2n+1)|D|$ bits of communication and $4n(2c+1)$ applications of a hash function for the sender and $2n(2c+1)$ applications for the receiver. Then κ OT_1^2 protocols can be implemented using the construction of [47] with low amortized complexity, where the sender performs $2+\kappa$ and the receiver performs 2κ modular exponentiations with the communication of $2\kappa^2$ bits and κ public keys. The OT protocols can be performed during the offline

stage, while the additional communication takes place once the inputs are known.

Further reducing online communication. If transmitting $2m$ ciphertexts during the online stage of the protocol (which amounts to a few hundred KB for our set of parameters) constitutes a burden, this communication can be performed at the offline stage before the protocol begins. This can be achieved using the technique of [49], where the client transmits $2m$ encryptions of randomly chosen bits $u_1, \ldots, u_{2m}$ during the offline stage, and the online communication consists of $2m$ bits $v_1, \ldots, v_{2m}$. Each bit v_i corresponds to the XOR of the bit w_i that the client wants to use in the protocol with the previously communicated random bit u_i. After receiving the $2m$-bit correction string $w_1 \oplus u_1, \ldots, w_{2m} \oplus u_{2m}$, the server needs to compute encryption of w_i's using $Enc(u_i)$ and v_i, which is done by XORing u_i and v_i inside the encryption. Using $u_i \oplus v_i = u_i(1 - v_i) + (1 - u_i)v_i = u_i + v_i - 2u_iv_i$, we see that when $v_i = 0$, the server can simply set $Enc(w_i) = Enc(u_i)$, but when $v_i = 1$, the server will need to perform subtraction of (encrypted) u_i. While subtraction is usually one of the most expensive operations, note that because of our use of DGK encryption with short plaintexts the subtraction operations can be performed on a ciphertext significantly faster than using generic full-domain encryption schemes such as Paillier. The speed up is on the order of $k/n \approx 50$, where $k \geq 1024$ is the security parameter for a public-key encryption scheme and $n = \lceil \log m \rceil + \ell = 20$ is the length of the values we operate on. Furthermore, this entire computation can be completely removed from the online stage if, upon the receipt of $Enc(u_i)$, the server computes $Enc(1 - u_i)$ during the offline stage. Then when the protocol begins, the server sets either $Enc(w_i) = Enc(u_i)$ or $Enc(w_i) = Enc(1 - u_i)$ depending on the bit v_i it receives.

4.4 Security Analysis

Security of the iris protocol relies on the security of the underlying building blocks. In particular, we need to assume that (i) the DGK encryption scheme is semantically secure (which was shown under a hardness assumption that uses subgroups of an RSA modulus [21, 20]); (ii) garbled circuit evaluation is secure (which was shown in [41], but the version we use [40] relies on a hash function which is assumed to be correlation robust or otherwise modeled as a random oracle); and (iii) the oblivious transfer is secure as well (to achieve this, techniques of [35] require the hash function to be correlation robust and the use of a pseudo-random number generator, while techniques of [47] model the hash functions as a random oracle and use the computational Diffie-Hellman (CDH) assumption). Therefore, assuming the security of the DGK encryption, CDH, and using the random oracle model for hash functions is sufficient for our approach.

To show the security of the protocol, we sketch how to simulate the view of each party using its inputs and outputs alone. If such simulation is indistinguishable from the real execution of the protocol, for semi-honest parties this implies that the protocol does not reveal any unintended information to the participants (i.e., they learn only the output and what can be deduced from their respective inputs and outputs).

First, consider the client C. The client's input consists of its biometric template X, $M(X)$ and the private key, and its outputs consists of a bit b for each record in S's database D. A simulator that is given these values simulates C's view by sending encrypted bits of C's input to the server as prescribed in step 1 of the protocol. It then simulates the messages received by the client in step 3(a).iii using encryptions of two randomly chosen strings r_C^j and t_C^j of length n. The simulator next creates a garbled circuit for the computation given in step 3(b) that, on input client's r_C^j's and t_C^j's computes bit b, sends the circuit to the client, and simulates the OT. Note that the simulator can set the other party's inputs in such a way that the computation results in bit b. It is clear that given secure implementation of garbled circuit evaluation in the real protocol, the client cannot distinguish simulation from real protocol execution. Furthermore, the values that C recovers in step 3(a).iv of the protocol are distributed identically to the values used in the real protocol execution that uses DGK encryption (and they are statistically indistinguishable when other encryption schemes are used).

Now consider the server's view. The server has its database D consisting of Y, $M(Y)$ and the threshold

Table 1: Breakdown of the performance of the iris identification protocol.

	Setup	Offline		
		Encryption	Circuit	Total
Server	$c = 5$	1398 msec + 71 msec/rec	1780 msec + 8.5 msec/rec	3178 msec + 79.5 msec/rec
	$c = 0$	1398 msec + 6.5 msec/rec	1457 msec + 0.75 msec/rec	2855 msec + 7.25 msec/rec
	$c = 5$ with [50]	131.37 sec + 780 msec/rec	1780 msec + 8.5 msec/rec	131.37 sec + 993.5 msec/rec
Client	$c = 5$	11.93 sec	1693 msec + 3.39 msec/rec	13.62 sec + 3.39 msec/rec
	$c = 0$	11.93 sec	1055 msec + 0.34 msec/rec	12.99 sec + 0.34 msec/rec
	$c = 5$ with [50]	161.37 sec	1693 msec + 3.39 msec/rec	163.06 sec + 3.39 msec/rec
Comm	$c = 5$	512KB	11.6KB + 22.1KB/rec	524KB + 22.1KB/rec
	$c = 0$	512KB	11.6KB + 2KB/rec	524KB + 2KB/rec
	$c = 5$ with [50]	1024KB	11.6KB + 22.1KB/rec	1036KB + 22.1KB/rec

	Setup	Online		
		Encryption	Circuit	Total
Server	$c = 5$	108 msec + 148 msec/rec	1.25 msec/rec	89 msec + 149.25 msec/rec
	$c = 0$	108 msec + 13.6 msec/rec	0.11 msec/rec	89 msec + 13.71 msec/rec
	$c = 5$ with [50]	427 msec + 586 msec/rec	1.25 msec/rec	427 msec + 587.25 msec/rec
Client	$c = 5$	20 msec/rec	2.61 msec/rec	22.61 msec/rec
	$c = 0$	1.8 msec/rec	0.22 msec/rec	2.02 msec/rec
	$c = 5$ with [50]	197 msec/rec	2.61 msec/rec	199.61 msec/rec
Comm	$c = 5$	0.5 KB + 2.7 KB/rec	17.2 KB/rec	0.5 KB + 19.9 KB/rec
	$c = 0$	0.5 KB + 0.2 KB/rec	1.6 KB/rec	0.5 KB + 1.8 KB/rec
	$c = 5$ with [50]	0.5 KB + 5.5 KB/rec	17.2 KB/rec	0.5 KB + 22.7 KB/rec

T as the input and no output. In this case, a simulator with access to D first sends to S ciphertexts (as in step 1 of the protocol) that encrypt bits of its choice. For each $Y \in D$, S performs its computation in step 3(a) of the protocol and forms garbled circuits as specified in step 3(b). The server and the simulator engage in the OT protocol, where the simulator uses arbitrary bits as its input to the OT protocol and the server sends the key-value mapping for the output gate. It is clear that the server cannot distinguish the above interaction from the real protocol execution. In particular, due to semantic security of the encryption scheme S learns no information about the encrypted values and due to security of OT S learns no information about the values chosen by the simulator for the garbled circuit.

4.5 Implementation and Performance

The implementation of the secure iris identification protocol that we describe was performed in C using MIRACL library [34] for cryptographic operations. It also used the DGK encryption scheme with a 1024-bit modulus and another security parameter t set to 160, as suggested in [21, 20]. To illustrate the advantage of the tools the solution utilizes, we also give performance of selected experiments using Paillier encryption [50]. The Paillier encryption scheme was implemented using a 1024-bit modulus and a number of optimizations suggested in [50] for best performance. In particular, small generator $g = 2$ was used to achieve lower encryption time, and decryption is sped up using pre-computation and Chinese remainder computation (see [50], section 7 for more detail). The security parameters $k = 1024$ for public-key cryptography and $\kappa = 80$ for symmetric and statistical security are used for compatibility with experiments reported in other sources, while larger security parameters would be preferred today. The experiments were run on an Intel Core 2 Duo 2.13 GHz machine running Linux (kernel 2.6.35) with 3GB of RAM and *gcc* version 4.4.5.

Table 1 shows performance of the secure iris identification protocol and its components. The perfor-

mance was obtained using the following set of parameters: the size of iris code and mask $m = 2048$ (this value of m is used in commercial iris recognition software), 75% of bits are reliable in each iris code, and the length of (plaintext) values used in the protocol is 20 bits (i.e., $n = \lceil \log m \rceil + \ell = 20$ in Figure 1). All optimizations described earlier in this section were implemented. In the implementation, upon the receipt of client's data, the server precomputes all combinations for pairs of ciphertexts $b_i b_{i+1}$ in step 3(a).ii (one-time cost of the total of $4(m/2)$ modular multiplications) and all combinations of 4 elements $d_i d_{i+1} d_{i+2} d_{i+3}$ in step 3(a).iii (one-time cost of $11(m/4)$ modular multiplications). This cuts the server's time for processing each Y by more than a half. Furthermore, the constant overhead associated with the OT (circuit) can be reduced in terms of both communication and computation for both parties if public-key operations are implemented over elliptic curves.

The table shows performance using three different configurations: (i) the amount of rotation c was set to 5, (ii) no rotation was used by setting $c = 0$ (this is used when the images are well aligned, e.g., during supervised image acquisition or when simultaneously acquiring images of both eyes), and (iii) with $c = 5$ using Paillier encryption instead of DGK scheme. In the table, we divide the computation and communication into offline pre-computation and online protocol execution. No inputs are assumed to be known by any party at pre-computation time. Some of the overhead depends on the server's database size, in which case the computation and communication are indicated per record (using notation ''/rec''). The overhead associated with the part of the protocol that uses homomorphic encryption is shown separately from the overhead associated with garbled circuits. The offline and online computation for the part based on homomorphic encryption is computed as described in Section 4.3. For circuits, garbled circuit creation, communication, and some of OT is performed at the offline stage, while the rest of OT (as described in Section 4.3) and garbled circuit evaluation take place during the online protocol execution. We also note that while the table lists computational overhead of each party separately, the overall runtime for a single biometric comparison will be approximately the runtime of the server (which is typically faster than the client and is not expected to be the bottleneck of the protocol) and the runtime of the server. The reason is that computation is initially carried out on encrypted data by the server, followed by the OT between the client and the server, followed by the client evaluating the garbled circuit. When, however, the parties perform a number of biometric comparisons, the amortized time per record in the database is going to be lower (i.e., it is the maximum of the server's and client's time instead of their sum) because all records can be evaluated in parallel and the amount of communication is low.

As the table indicates, the design of the solution and the optimizations employed in it allow for a particularly efficient performance. In particular, comparison of two iris codes, which among other things includes computation of $2(2c+1) = 22$ Hamming distances (i.e., for the numerator and denominator in Equation 1) over 2048-bit biometric templates in encrypted form, is done in 0.15 sec. This is comparable in speed to the latest developments in other functionalities (e.g., [32, 15, 54], which can be used to compute the Hamming distance) and in part due to the use of efficient DGK encryption scheme and other optimizations. When iris images are well aligned and no rotation is necessary, our protocol requires only 14 msec online computation time and under 2KB of data to compare two biometric templates.

5 Secure Fingerprint Identification

Before proceeding with new techniques for fingerprint identification based on minutiae pairing, we first illustrate how the techniques given in this chapter for iris identification can be applied to other types of biometric computations such as FingerCodes. In particular, they can be used to improve the efficiency of the secure protocol for FingerCode identification in [5].

Input: C has biometric template $X = (x_1, \ldots, x_m)$ and DGK encryption key pair (pk, sk); S has a database D composed of biometric templates $Y = (y_1, \ldots, y_m)$.
Output: C learns what records in D resulted in match with X if any, i.e., it learns a bit as a result of comparison of X with each $Y \in D$.
Protocol steps:

1. C computes and sends to S encryptions $Enc(-2x_1), \ldots, Enc(-2x_m), Enc(\sum_{i=1}^{m} x_i^2)$.
2. For each $Y = (y_1, \ldots, y_m) \in D$, S and C perform in parallel:
   (a) S computes the encrypted distance d between X and Y as $d = Enc(\sum_{i=1}^{m} x_i^2) \cdot Enc(\sum_{i=1}^{m} y_i^2) \cdot \prod_{i=1}^{m} Enc(-2x_i)^{y_i} = Enc(\sum_{i=1}^{m}(x_i - y_i)^2)$, blinds it as $d' = d \cdot Enc(r_S)$, where $r_S \xleftarrow{R} \{0,1\}^n$, and sends d' to C.
   (b) C decrypts the value it receives and sets $r_C = Dec(d')$.
   (c) C and S engage in a secure protocol that computes $((r_C - r_S) \bmod 2^n) \overset{?}{<} T^2$ using garbled circuit evaluation. S creates the circuit and sends it to C along with the key-value mapping for the output gate. C obtains keys corresponding to its inputs from S using OT, evaluates the circuit, and learns the result.

Figure 2: Secure two-party protocol for FingerCode identification.

5.1 FingerCode Identification

The computation involved in FingerCode comparisons is very simple, which results in an extremely efficient privacy-preserving realization. We rewrite the computation in Equation 4 as $\sum_{i=1}^{m}(x_i - y_i)^2 = \sum_{i=1}^{m}(x_i)^2 + \sum_{i=1}^{m}(y_i)^2 - \sum_{i=1}^{m} 2x_i y_i < T^2$. In our protocol, the Euclidean distance is computed using homomorphic encryption, while the comparisons are performed using garbled circuits. The secure FingerCode protocol is given in Figure 2: The client contributes encryptions of $-2x_i$ and $\sum(x_i)^2$ to the computation, while the server contributes $\sum(y_i)^2$ and computes encryption of $-2x_i y_i$ from $-2x_i$. Note that by using $Enc(-2x_i)$ instead of $Enc(x_i)$, the server's work for each Y is reduced since negative values typically use significantly longer representations. The protocol in Figure 2 uses DGK encryption with the plaintext space of $[0, 2^n - 1]$. To be able to represent the Euclidean distance, we need to set $n = \lceil \log m \rceil + 2\ell + 1$, where ℓ is the bitlength of elements x_i and y_i. This implies that all computation on plaintexts is performed modulo 2^n; for instance, $2^n - 2x_i$ is used in step 1 to form $Enc(-2x_i)$. The circuit used in step 2(c) takes two n-bit values, adds them modulo 2^n, and compares the result to a constant as described in Section 4.3.

Finally, some of the computation can be performed offline: For the client it includes precomputing the random values used in the $m + 1$ ciphertexts it sends in step 1 (i.e., the computation of $h^r \bmod N$). For the server it includes precomputing $Enc(r_S)$, preparing a garbled circuit for each Y, and one-time computation of random values for $Enc(\sum_{i=1}^{m}(y_i)^2)$ since the reuse of ciphertexts in this case does not affect security. The client and the server also perform some of OT functionality prior to protocol initiation, as previously discussed.

In literature on FingerCodes, each fingerprint in the server's database is represented by c FingerCodes that correspond to different orientations of the same fingerprint, which improves the accuracy of comparison. Then if the client is entitled to receiving all matches within the c FingerCodes corresponding to the same fingerprint, our protocol in Figure 2 computes exactly this functionality. If, on the other hand, it is desirable to output only a single bit for all c instances of a fingerprint, it is easy to modify the circuit evaluated in step 2(c) of the protocol to compute the OR of the bits produced by the original c circuits.

Security. The security of this protocol is straightforward to show and we omit the details of the simulator from the current description. As before, by using only tools known to be secure and protecting the information at intermediate stages, neither the client nor the server learns information beyond what the protocol prescribes.

Table 2: Breakdown of the performance of the FingerCode identification protocol.

	Offline		
	Encryption	**Circuit**	**Total**
Server	3.6 msec + 3.9 msec/rec	1448 msec + 0.37 msec/rec	1451.6 msec + 4.3 msec/rec
Client	61 msec	1025 msec + 0.15 msec/rec	1086 msec + 0.15 msec/rec
Comm	0	11.6 KB + 1.26 KB/rec	11.6 KB + 1.26 KB/rec

	Online		
	Encryption	**Circuit**	**Total**
Server	0.22 msec + 1.37 msec/rec	0.05 msec/rec	0.22 msec + 1.42 msec/rec
Client	4.7 msec + 0.92 msec/rec	0.16 msec/rec	4.7 msec + 1.08 msec/rec
Comm	2.12 KB + 0.12 KB/rec	0.74 KB/rec	2.12 KB + 0.86 KB/rec

Implementation and performance. The FingerCode parameters can range as $m = 16$-640, $\ell = 4$-8, and $c = 5$. The implementation we report uses parameters $m = 16$ and $\ell = 7$ and therefore $n = 19$. The performance of the secure FingerCode identification protocol is given in Table 2. No inputs (X or Y) are assumed to be known at the offline stage when the parties compute (among other things) randomization values of the ciphertexts. For that reason, a small fixed cost is inquired in the beginning of the protocol to finish forming the ciphertexts using the inputs. We also note that, based on additional experiments, by using Paillier encryption instead of DGK encryption, the server's online work increases by an order of magnitude, even if packing of multiple elements into a single ciphertext is used with Paillier encryption.

It is evident that the overhead reported in the table is minimal and the protocol is suitable for processing fingerprint data in real time. For example, for a database of 320 records (64 fingerprints with 5 FingerCodes each), client's online work is 0.35 sec and the server's online work is 0.45 sec, with online communication of 277KB. As can be seen from these results, computation is no longer the bottleneck and this secure two-party protocol can be carried out very efficiently.

5.2 Minutia-Based Fingerprint Identification

We next present a secure protocol for minutia-based fingerprint identification. It preserves the high-level idea of using homomorphic encryption for computing the distance between minutia points and garbled circuit evaluation for comparisons, but introduces a number of new techniques. At high-level, computing the pairing between minutiae of fingerprints $X = \langle (x_1, y_1, \alpha_1), \ldots, (x_{m_X}, y_{m_X}, \alpha_{m_X}) \rangle$ and $Y = \langle (x'_1, y'_1, \alpha'_1), \ldots, (x'_{m_Y}, y'_{m_Y}, \alpha'_{m_Y}) \rangle$ based on minimum distances between the points proceeds in iterations as follows. C and S maintain an m_Y-bit array M, the i-th bit of which indicates whether minutia Y_i has been marked or not. Initially, all bits of M are set to 0. For $i = 1, \ldots, m_X$, perform:

1. Compute the set Z of minutiae from Y matching X_i that have not been marked, i.e., $Z = \{Y_j \mid mm(X_i, Y_j) \text{ and } M[j] = 0\}$.
2. Compute the minutia Y_k (if any) from Z with the minimum (spatial) distance from X_i, and set $M[k] = 1$.

To preserve secrecy of the data, each bit of the array M is maintained by C and S in XOR-split form, i.e., C stores $M_C[i]$ and S stores $M_S[i]$ such that $M[i] = M_C[i] \oplus M_S[i]$. During each iteration of the computation, at the end of step 2 above, C and S obtain XOR-shares of an array A that has bit k set to 1 and all other bits set to 0 (and all bits are set to 0 if no pairing for X_i exists). Both C and S update their share of M by XORing the share of A that they received with the current share of M. This ensures that the array M is properly maintained.

Input: C has biometric template $X = \langle (x_1, y_1, \alpha_1), \ldots, (x_{m_X}, y_{m_X}, \alpha_{m_X}) \rangle$ and DGK encryption key pairs (pk_1, sk_1) and (pk_2, sk_2); S has a database D composed of biometric templates in the form $Y = \langle (x'_1, y'_1, \alpha'_1), \ldots, (x'_{m_Y}, y'_{m_Y}, \alpha'_{m_Y}) \rangle$.
Output: C learns what records in D resulted in match with X if any, i.e., it learns a bit as a result of comparison of X with each $Y \in D$.
Protocol steps:

1. C computes encryptions $\langle a_{i1}, a_{i2}, a_{i3}, a_{i4} \rangle = \langle Enc_{pk1}(-2x_i), Enc_{pk_1}(-2y_i), Enc_{pk_1}(x_i^2 + y_i^2), Enc_{pk_2}(-\alpha_i) \rangle$ for each $i = 1, \ldots, m_X$ and sends them to S.
2. For each $Y = \langle (x'_1, y'_1, \alpha'_1), \ldots, (x'_{m_Y}, y'_{m_Y}, \alpha'_{m_Y}) \rangle \in D$, S and C perform in parallel:
   (a) S and C setup m_Y-bit vector M, where initially S's and C's shares M_S and M_C, respectively, are set to all 0's.
   (b) For $i = 1, \ldots, m_X$ S and C perform the following computation:
      i. S computes the encrypted spatial distance s_j between X_i and each Y_j in Y as $s_j = (a_{i1})^{x'_j} \cdot (a_{i2})^{y'_j} \cdot a_{i3} \cdot Enc_{pk_1}((x'_j)^2 + (y'_j)^2)$ and encrypted directional distance as $d_j = (a_{i4})^{\alpha'_j}$. S blinds all pairs as $s'_j = s_j \cdot Enc(r_S^j)$, where $r_S^j \overset{R}{\leftarrow} \{0,1\}^{2\ell+2}$ and $d'_j = d_j \cdot Enc(t_S^j)$ where $t_S^j \overset{R}{\leftarrow} \mathbb{Z}_{360}$ and sends s'_j, d'_j to C.
      ii. C decrypts received pairs for all $j = 1, \ldots, m_Y$ and sets $r_C^j = Dec_{sk_1}(s'_j)$ and $t_C^j = Dec_{sk_2}(d'_j)$.
      iii. C and S engage in garbled circuit evaluation, where S inputs the bits of M_S and $-r_S^j \pmod{2^{2\ell+2}}, -t_S^j \pmod{360}$ for $j = 1, \ldots, m_Y$, C inputs the bits of M_C and r_C^j, t_C^j for $j = 1, \ldots, m_Y$, S learns m_Y-bit A_S, and C learns m_Y-bit A_C. The vector $A = A_S \oplus A_C$ has at most one bit set which indicates the index of the mate of minutia X_i in Y.
      iv. S updates its M_S as $M_S = M_S \oplus A_S$, and C updates its M_C as $M_C = M_C \oplus A_C$.
   (c) C and S engage in the garbled circuit evaluation where, on input M_S from S and M_C from C, C learns the bit corresponding to the computation $||M_S \oplus M_S|| \overset{?}{<} T$.

Figure 3: Secure two-party protocol for minutia-based fingerprint identification.

In the beginning of the protocol the client sends information about its fingerprint X. For best performance, the solution utilizes DGK encryption with two pairs of keys. The first pair (pk_1, sk_1) is used for encrypting spatial coordinates x_i, y_i and computing Euclidean distance between points, and the second pair (pk_2, sk_2) is used for encrypting orientation information α_i and directional difference. Therefore, we set the plaintext space $u = 2^{2\ell+2}$ in pk_1, where ℓ is the bitlength of coordinates x_i, y_i, and $u = 360$ in pk_2. This implies that computing $\alpha'_j - \alpha_i$ on encrypted values will automatically result in the value being reduced modulo 360, which simplifies computation with the directional difference in this form. Also note that, while decryption in the DGK encryption scheme involves solving the discrete logarithm, when $u = 360$ this can be achieved at low cost using Pohlig-Hellman algorithm because 360 has only small factors.

Our secure fingerprint identification protocol is given in Figure 3. At iteration i, after computing the distances in encrypted form (step 2(b).i) and decrypting them in a secret-shared form (step 2(b).ii), the parties engage in garbled circuit evaluation using a circuit that performs the main computation and produces an m_Y-bit vector A with at most one bit set to one indicating the position of the mate of minutia X_i. This (optimized) circuit is the most involved part of the protocol and is discussed in detail below. At the end of each iteration the vector M is updated with the output of the circuit. And after all iterations have been performed, the rest of the protocol consists of counting the number of marked elements in M and comparing that number to the threshold T. This is done using an additional garbled circuit, where the client learns the output bit.

Note that the protocol requires that both parties know the number of minutiae in client's X and server's Ys, which is assumed not to leak information about the fingerprints themselves. While biometric images of similar quality are expected to have similar numbers of minutiae, if for the purposes of this computation m_X and m_Y are considered to be sensitive information, the fingerprints can be slightly padded to always use the same number m of minutia points. This can be achieved by agreeing on a fixed m and inserting fake

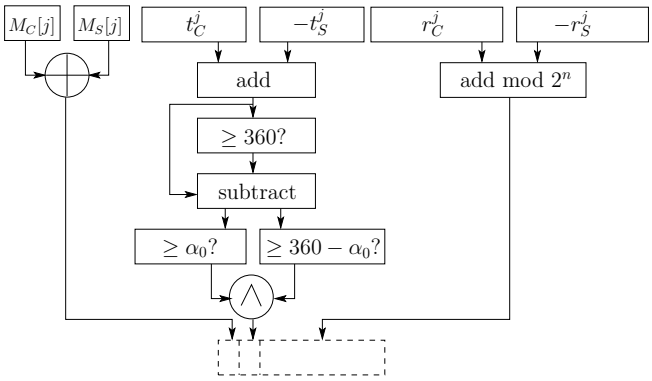

Figure 4: Component of circuit in fingerprint identification protocol performed for each value of $j \in [1, m_Y]$.

elements into each fingerprint until its size becomes m. The fake elements should not affect the result of the computation, which means that the fake elements of client's X should not match either the original or fake elements of any Y. The easiest way to ensure this is by setting fake x_i in X to its maximum value plus d_0 and by setting fake x'_j in each Y to its maximum value plus $2d_0$. This slightly increases the range of values that spatial distances between minutia points can take, but is not likely to result in additional overhead due to the increased space (i.e., the overhead can increase only when the bitlength needed to represent the distances grows).

We design the circuit in step 2(b).iii of the protocol to minimize the number of comparisons. In particular, each directional difference $\alpha'_j - \alpha_i$ is compared to the threshold α_0 in the beginning, and if it exceeds the threshold, the corresponding distance between X_i and Y_j is modified so that it will not be chosen as the minimum. This is done by prepending the resulting bit of computation $((\alpha'_j - \alpha_i) \geq \alpha_0) \wedge ((\alpha'_j - \alpha_i) \leq (360 - \alpha_0))$ to the spatial distance between X_i and Y_j (as the most significant bit). The same technique is used to ensure that marked minutiae from Y are not selected as well. What remains to do is to verify what spatial distances fall below the threshold and compute the minimum of such values. In the (oblivious) garbled circuit, instead of first comparing each distance to the threshold and then computing the minimum of (possibly modified) distances, the solution directly computes the minimum and then compares the minimum to the threshold. This reduces the number of distance comparisons from $2m_Y - 1$ to m_Y. The two previously prepended bits are preserved throughout the comparisons, and the final result will have no mate for X_i selected if the computed minimum (squared) distance is not below the threshold $(d_0)^2$.

Both the computation of the minimum and creation of vector A require the use of multiplexers in the circuit. In particular, after comparing two values a_1 and a_2 one type of multiplexer used in our circuit chooses either the bits of a_1 or a_2 based on the resulting bit of the comparison. This permits the computation of the minimum in a hierarchical manner using a small number of non-XOR gates as described in [39]. We also use multiplexers to collect information about A throughout the circuit. In particular, after a single comparison of distances a_1 and a_2, the portion of A corresponding to these two bits will be chosen to be either 01 or 10. Suppose that after comparing a_1 and a_2 this value is 01 and after comparing a_3 and a_4 the value is 10. Then after performing the comparison of $\min(a_1, a_2)$ and $\min(a_3, a_4)$ either 0100 or 0010 will be chosen as the current portion of A. This process continues until the overall minimum and the entire A is computed. This value of A will have a single bit set to 1, and after the final comparison of the minimum with the threshold A will either remain unchanged or will be reset to contain all 0s.

Figure 4 shows the initial computation in the circuit performed for each value of j, where $n = 2\ell + 2$, and Figure 5 shows the computation of the minimum and the output for a toy example of $m_Y = 4$. In Figure 4, after adding t_C^j and $-t_S^j \pmod{360}$ together, the sum is compared to 360. If it exceeds the value, 360 is subtracted from the sum (in our concrete realization the subtracted value is bitwise AND of the outcome of

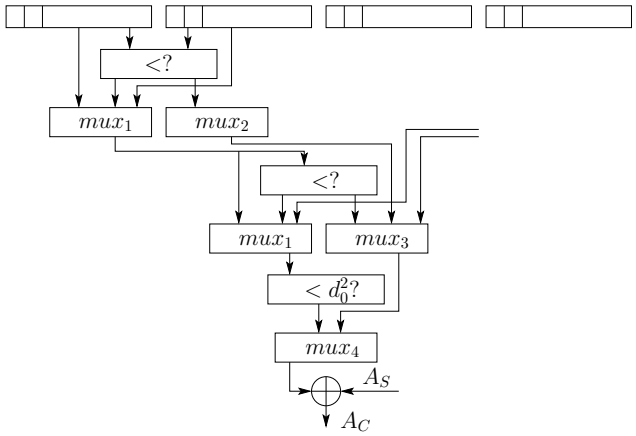

Figure 5: Computation of minimum and its index in circuit of fingerprint identification.

the comparison and each bit of the binary representation of 360). Finally, the resulting value is compared to two thresholds and the result is prepended to the spatial distance $r_C^j - r_S^j$. In Figure 4, multiplexer mux_1 chooses the smaller value based on the result of the comparison, mux_2 chooses either 01 or 10 based on the result of the comparison, mux_3 chooses a 4-bit string based on its inputs from two multiplexers mux_2 and the outcome of another comparison, and mux_4 chooses either its input from mux_3 or a zero string based on the result of the final comparison. The server (circuit creator) supplies a stream of random bits A_S to the circuit, and the client learns the outcome of the XOR of that stream and the output of the last multiplexer.

Precomputation. Precomputation in this protocol takes a similar form to that in the FingerCode protocol. Namely, the random values ($h^r \bmod N$) in the ciphertexts are precomputed and the server chooses all r_S^j and t_S^j in advance and encrypts them. Furthermore, omitting randomness in the encrypted values $Enc((x'_j)^2 + (y'_j)^2)$ at the server side does not compromise security and thus the server skips precomputing the randomization component of such ciphertexts (and assumes $h^r \bmod N = 1$) for each j and each $Y \in D$, resulting in substantial savings. In addition, all garbled circuits are created and transferred in advance, as well as the expensive portion of the OT is performed in advance. Note that it is sufficient to have two input wires to implement all constants in the circuit such as 360, α_0, d_0^2, inputs to the multiplexers, etc.

Security. As before, it is easy to show that the protocol is secure, where the simulator relies on the security of the encryption scheme, garbled circuits, and OT.

Implementation and performance. To show performance of the protocol, we use a grid of size 250×250 for minutiae coordinates, which means that each $x_i, y_i \in [0, 249]$ and ℓ is set to 8. In the experiments that follow, $m = m_X = m_Y$ with two values of 20 and 32 minutiae per fingerprint. It is clear that the protocol incurs cost quadratic in m and is expected to have higher overhead than two previous iris and FingerCode protocols. Table 3 shows performance of our secure minutia-based fingerprint comparisons. The online work is dominated by $2m^2$ decryptions at the client side and adds up to 0.73 sec per fingerprint comparison for $m = 20$ and 1.88 sec for $m = 32$. The circuit evaluated by the client in step 2(b).iii of the protocol has 2372 non-XOR and 8836 total gates for $m = 20$ and 3820 non-XOR and 14212 total gates for $m = 32$. It is evaluated m times by the client for each Y. The circuit evaluated by the client in step 2(c) of the protocol has 39 non-XOR and 153 total gates for $m = 20$ and 63 non-XOR and 246 total gates for $m = 32$. It is evaluated once for each Y.

We also would like to mention that a protocol solely based on garbled circuit evaluation for this type of computation is likely to result in comparable or possibly even faster performance due to recent advances in

Table 3: Breakdown of the performance of the fingerprint identification protocol.

	Setup	Offline		
		Encryption	Circuit	Total
Server	$m = 20$	72 msec + 2990 msec/rec	1868 msec + 1159 msec/rec	1940 msec + 4149 msec/rec
	$m = 32$	114 msec + 7682 msec/rec	2114 msec + 2117 msec/rec	2228 msec + 9799 msec/rec
Client	$m = 20$	288 msec	1866 msec + 212 msec/rec	2154 msec + 212 msec/rec
	$m = 32$	460 msec	2380 msec + 552 msec/rec	2840 msec + 552 msec/rec
Comm	$m = 20$	0	11.6KB + 83KB/rec	11.6KB + 83KB/rec
	$m = 32$	0	11.6KB + 133KB/rec	11.6KB + 133KB/rec

	Setup	Online		
		Encryption	Circuit	Total
Server	$m = 20$	3.6 msec + 100 msec/rec	30 msec/rec	3.6 msec + 130 msec/rec
	$m = 32$	6 msec + 262 msec/rec	77 msec/rec	6 msec + 339 msec/rec
Client	$m = 20$	15 msec + 580 msec/rec	145 msec/rec	15 msec + 725 msec/rec
	$m = 32$	25 msec + 1502 msec/rec	374 msec/rec	25 msec + 1876 msec/rec
Comm	$m = 20$	10KB + 100KB/rec	22.3KB/rec	10KB + 122.3KB/rec
	$m = 32$	16KB + 256KB/rec	38.2KB/rec	16KB + 294.2KB/rec

the speed of garbled circuit evaluation and OT extensions (such as [7] and [38, 2], respectively). To realize that, the circuit would need to perform additional $2m^2$ multiplications (as well as additional additions and subtractions) per Y, with the additional number of gates exceeding the current number of gates in the circuits. This means that using the techniques that we implement the offline work associated with circuit construction (per Y) will increase, but the online communication should decrease.

6 Summary of Design Principles and Results

The protocol design presented in this chapter suggests certain principles that lead to an efficient implementation of a privacy-preserving protocol for biometric identification. First, notice that in the computation described in this chapter, as well as in other literature, first a distance between biometric template X and each biometric template Y in the database is computed followed by a comparison operation. The comparison can be performed to either (i) determine whether the distance $dist(X, Y)$ is below a certain threshold (where the threshold can be specific to each Y or fixed for all Y) or (ii) determine whether the minimum of all distances $dist(X, Y)$ is below a certain threshold. In both cases an equivalent number of comparisons is performed. Several existing efficient protocols compute the distance function using homomorphic encryption, but then resort to a different technique for the comparisons. Therefore, the client first communicates its encrypted biometric template X to the server, the server next computes the distances, and both the client and the server are involved in the comparison protocol. We thus obtain the following:

1. *Representation of client's biometric data matters.* The server's work for processing each record in its database can be significantly reduced if the client's data is provided in the form that optimizes server's computation (for instance, computing $Enc(-a)$ from $Enc(a)$ could be one of the most expensive operations). This one-time cost at the client's side has far-reaching consequences for the performance of the overall protocol.

2. *Operations that manipulate bits are the fastest outside encryption.* Any protocol for biometric identification is expected to use comparisons. Despite recent advances in the techniques for carrying out secure comparisons over encrypted data which make them practical (e.g., [21]), garbled circuit evaluation is better suited for a large volume of such operations. Furthermore, when the range of values

being compared is small and many comparisons are necessary, additional techniques such as OT can be utilized at low cost [49].

3. *A substantial speedup can be seen from proper tuning of encryption tools.* Privacy-preserving protocols that rely on homomorphic encryption can benefit immensely from a wise choice of encryption scheme and its usage. Traditionally, packing was used to reduce overhead of privacy-preserving protocols including asymptotic complexity (see, e.g., [44] for an example). When computation is carried out on integers of small size, alternative encryption schemes such as DGK or additively homomorphic ElGamal implemented over elliptic curves can significantly improve performance. The performance that the solutions presented in this chapter achieve would not be possible without the right choice of encryption schemes.

Using these principles and a number of new techniques, in this chapter we demonstrate secure protocols for iris and fingerprint identification that use standard biometric recognition algorithms. The optimization techniques employed in this chapter allow for fast performance of three secure biometric identification protocols:

- One of the first privacy-preserving two-party protocols for iris codes using current biometric recognition algorithms. Despite the length of iris codes' representation and complexity of their processing, the protocol we present allows a secure comparison between two biometric templates to be performed in 0.15 second with communication of under 18KB. Furthermore, when the iris codes are known to be well-aligned and their rotation is not necessary, the overhead decreases by an order of magnitude to 14 msec computation and 2KB communication per comparison.
- A privacy-preserving and extremely efficient two-party protocol for FingerCodes used for low-cost fingerprint recognition. Comparing two fingerprints requires approximately 1 msec of computation, allowing thousands of biometric templates to be processed in a matter of seconds. Communication overhead is also very modest with less than 1KB per biometric comparison.
- Secure fingerprint recognition based on minutiae pairings that utilizes most complex algorithms over unordered sets with spatial and directional differences. The implementation results suggest that such secure fingerprint identification can be performed using approximately 1 second per record.

7 Further Reading

Most of the material presented in this chapter appeared in [10]. Two privacy-preserving approaches for FingerCodes (with higher overhead) are available in [5] and in [33]. Minutia-based fingerprint matching (also with higher overhead) is available in [55]. Additionally, some publications (e.g., [25]) propose alternative mechanisms for privacy-preserving authentication (as opposed to identification) without using standard algorithms for comparing two biometric templates.

A number of publications [24, 53, 49] target the problem of privacy-preserving face recognition. The first two of these [24, 53] build solutions based on the Eigenfaces algorithm (where [53] improves the performance of the technique in [24]), while [49] designs a new face recognition algorithm together with its privacy-preserving realization called SciFi. The design targeted to simultaneously address robustness of the face recognition algorithm to different viewing conditions and efficiency when used for secure computation. Several papers have since improved on the techniques used in SciFi [32, 15, 54].

The problem of privacy-preserving iris matching has been addressed in [43] and [16] using garbled circuits. Luo et al. [43] reduce computational cost of iris matching by using a common mask between the two protocol participants, although at the cost of a slight increase of false accept and false reject rates. Bringer et al. [16] use filtering techniques to improve performance of secure iris matching.

There are also publications that treat biometric authentication with privacy protection without implementing an exact algorithm used to compare a specific biometric modality. For example, [13] describes a biometric authentication mechanism where the Hamming distance is used as the distance metric and the authentication server is composed of three entities that must not collude. Consequently, [4] extends that work with a classifier.

General secure multi-party computation techniques can also be used for secure biometric identification, and literature on this topic is extensive. Starting from the seminal work on garbled circuit evaluation [59, 28], it has been known that any function can be securely evaluated by representing it as a boolean circuit. Similar results are also known for securely evaluating any function using secret sharing techniques (e.g., [52]) or homomorphic encryption (e.g., [19]). In the last several years a number of tools have been developed for automatically creating a secure protocol from a function description written in a high-level language. Examples include Fairplay [45], VIFF [22], TASTY [29], PICCO [60], and others. It is, however, usually the case that custom optimized protocols constructed for specific applications outperform solutions based on general techniques. Such custom solutions are known for a wide range of application (e.g., set operations, DNA matching, k-means clustering, etc.), and this chapter focuses on secure biometric identification using iris codes and fingerprints. Furthermore, some of the optimizations presented in this chapter can find their uses in protocol design for other applications, as well as general compilers and tools such as TASTY [29].

An overview of privacy-preserving biometric identification is presented in [14].

References

[1] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele. Secure computation on floating point numbers. In *Network and Distributed System Security Symposuim (NDSS)*, 2013.

[2] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM Conference on Computer and Communications Security (CCS)*, pages 535--548, 2013.

[3] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103--114, 2004.

[4] M. Barbosa, T. Brouard, S. Cauchie, and S. de Sousa. Secure biometric authentication with improved accuracy. In *Australasian conference on Information Security and Privacy (ACISP)*, pages 21--36, 2008.

[5] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-preserving fingercode authentication. In *ACM Workshop on Multimedia and Security (MM&Sec)*, pages 231--240, 2010.

[6] N. Barzegar and M. Moin. A new user dependent iris recognition system based on an area preserving pointwise level set segmentation approach. *EURASIP Journal on Advances in Signal Processing*, pages 1--13, 2009.

[7] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE Symposium on Security and Privacy*, pages 478--492, 2013.

[8] M. Blanton. Empirical evaluation of secure two-party computation models. Technical Report TR 2005-58, CERIAS, Purdue University, 2005.

[9] M. Blanton and M. Aliasgari. Secure computation of biometric matching. Technical Report 2009-03, Department of Computer Science and Engineering, University of Notre Dame, 2009.

[10] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *European Symposium on Research in Computer Security (ESORICS)*, pages 190--209, 2011.

[11] M. Blanton, A. Steele, and M. Aliasgari. Data-oblivious graph algorithms for secure computation and outsourcing. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 183--194, 2013.

[12] D. Bogdanov, M. Niitsoo, T. Toft, and J. Willemson. High-performance secure multi-party computation for data mining applications. *International Journal of Information Security*, 11(6):403--418, 2012.

[13] J. Bringer, H. Chabanne, M. Izabachene, D. Pointcheval, Q. Tang, and S. Zimmer. An application of the Goldwasser-Micali cryptosystem to biometric authentication. In *Australasian conference on Information Security and Privacy (ACISP)*, volume 4586 of *LNCS*, pages 96--106, 2007.

[14] J. Bringer, H. Chabanne, and A. Patey. Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends. *IEEE Signal Processing Magazine*, 30(2):42--52, 2013.

[15] J. Bringer, H. Chabanne, and A. Patey. SHADE: Secure Hamming distance computation from oblivious transfer. In *Financial Cryptography Workshops*, volume 7862 of *LNCS*, pages 164--176. Springer, 2013.

[16] J. Bringer, M. Favre, H. Chabanne, and A. Patey. Faster secure computation for biometric identification using filtering. In *IAPR International Conference on Biometrics (ICB)*, pages 257--264, 2012.

[17] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *ACM Conference on Computer and Communications Security (CCS)*, pages 486--497, 2007.

[18] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *Financial Cryptography and Data Security*, pages 35--50, 2010.

[19] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology - EUROCRYPT*, pages 280--300, 2001.

[20] I. Damgård, M. Geisler, and M. Krøigård. A correction to efficient and secure comparison for on-line auctions. Cryptology ePrint Archive, Report 2008/321, 2008.

[21] I. Damgård, M. Geisler, and M. Krøigård. Homomorphic encryption and secure comparison. *Journal of Applied Cryptology*, 1(1):22--31, 2008.

[22] I. Damgård, M. Geisler, and M. Krøigård. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography (PKC)*, pages 160--179, 2009.

[23] J. Daugman. How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):21--30, 2004.

[24] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enchancing Technologies Symposium (PETS)*, pages 235--253, 2009.

[25] Q. Feng, F. Su, and A. Cai. Privacy-preserving authentication using fingerprint. *International Journal of Innovative Computing, Information and Control*, 8(11):8001--8018, 2012.

[26] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[27] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[28] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, pages 218--229, 1987.

[29] W. Henecka, S. Kogl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. In *ACM Conference on Computer and Communications Security (CCS)*, pages 451--462, 2010.

[30] T. Hoens, M. Blanton, and N. Chawla. A private and reliable recommendation system using a social network. In *IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*, pages 816--825, 2010.

[31] K. Hollingsworth, K. Bowyer, and P. Flynn. The best bits in an iris code. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):964--973, June 2009.

[32] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, 2011.

[33] Y. Huang, L. Malka, D. Evans, and J. Katz. Efficient privacy-preserving biometric identification. In *Network and Distributed System Security Symposuim (NDSS)*, 2011.

[34] Multiprecision Integer and Rational Arithmetic C/C++ Library. http://www.shamus.ie/.

[35] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious tranfers efficiently. In *Advances in Cryptology - CRYPTO*, pages 145--161, 2003.

[36] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. *IEEE Transactions on Image Processing*, 9(5):846--859, 2000.

[37] T.-Y. Jea and V. Govindaraju. A minutia-based partial fingerprint recognition system. *Pattern Recognition*, 38(10):1672--1684, 2005.

[38] V. Kolesnikov and R. Kumaresan. Improved OT extension for transferring short secrets. In *CRYPTO*, pages 54--70, 2013.

[39] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security (CANS)*, pages 1--20, 2009.

[40] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 486--498, 2008.

[41] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161--188, 2009.

[42] Y. Lindell, B. Pinkas, and N. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *Security and Cryptography for Networks (SCN)*, pages 2--20, 2008.

[43] Y. Luo, S.-C. Cheung, T. Pignata, R. Lazzeretti, and M. Barni. An efficient protocol for private iris-code matching by means of garbled circuits. In *IEEE International Conference on Image Processing (ICIP)*, pages 2653--2656, 2012.

[44] K. Nissim M. Freedman and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT*, pages 1--19, 2004.

[45] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - a secure two-party computation system. In *USENIX Security Symposium*, pages 287--302, 2004.

[46] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar. *Hanbook of Fingerprint Recognition*. Springer, second edition, 2009.

[47] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 448--457, 2001.

[48] U.S. DHS Office of Biometric Identity Management. http://www.dhs.gov/obim.

[49] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. SCiFI - A system for secure face identification. In *IEEE Symposium on Security and Privacy*, pages 239--254, 2010.

[50] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223--238, 1999.

[51] B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. In *Advances in Cryptology - ASIACRYPT*, volume 5912 of *LNCS*, pages 250--267, 2009.

[52] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, pages 73--85, 1989.

[53] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology (ICISC)*, pages 229--244, 2009.

[54] T. Schneider and M. Zohner. GMW vs. Yao? Efficient secure two-party computation with low depth circuits. In *Financial Cryptography and Data Security*, volume 7859 of *LNCS*, pages 275--292, 2013.

[55] S. Shahandashti, R. Safani-Naini, and P. Ogunbona. Private fingerprint matching. In *Australasian Conference on Information Security and Privacy (ACISP)*, pages 426--433, 2012.

[56] The Corbett Report. India fingerprints, iris scanning over one billion people. http://www.corbettreport.com/india-fingerprinting-iris-scanning-over-one-billion-people/.

[57] UAE Iris Collection. http://www.cl.cam.ac.uk/~jgd1000/UAEdeployment.pdf.

[58] C. Wang, M. Gavrilova, Y. Luo, and J. Rokne. An efficient algorithm for fingerprint matching. In *International Conference on Pattern Recognition (ICPR)*, pages 1034--1037, 2006.

[59] A. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162--167, 1986.

[60] Y. Zhang, A. Steele, and M. Blanton. PICCO: A general-purpose compiler for private distributed computation. In *ACM Conference on Computer and Communications Security (CCS)*, pages 813--826, 2013.