BEAT

Biometrics Evaluation and Testing

http://www.beat-eu.org/

Funded under the 7th FP (Seventh Framework Programme)

Theme SEC-2011.5.1-1
[Evaluation of identification technologies, including Biometrics]

D5.3: Description of Advanced Privacy-Preservation Techniques

Due date: 30/04/2014 **Submission date:** 30/04/2014
Project start date: 01/03/2012 **Duration:** 48 months
WP Manager: C. Karabat (TUBITAK) **Revision:** 0

Author(s): A. Abidin (Chalmers), J. Bringer (MORPHO), S. Faust (EPFL), C. Karabat (TUBITAK), A. Mitrokotsa (Chalmers), R. Reyhanitabar (EPFL) and S. Vaudenay (EPFL)

Project funded by the European Commission in the 7th Framework Programme (2008-2010)		
Dissemination Level		
PU	Public	Yes
RE	Restricted to a group specified by the consortium (includes Commission Services)	No
CO	Confidential, only for members of the consortium (includes Commission Services)	No

D5.3: Description of Advanced Privacy-Preservation Techniques

Abstract:

This deliverable provides an overview of existing state-of-the-art advances in privacy-preserving biometric authentication schemes. More precisely, we go a step beyond the privacy-preserving biometric authentication schemes that have been described in deliverable D5.1 and focus on authentication schemes that rely on secure multi-party computation. We describe how oblivious transfer, garbled circuits and homomorphic encryption as well as fuzzy extractors may be employed to guarantee privacy-preservation in the biometric authentication problem and we list some of the latest privacy-preserving biometric authentication protocols that rely on these approaches.

Contents

1 Introduction

Efficient biometric authentication is a valuable tool with many important applications. There have been significant technological advances in biometrics and an increase in their use as a reliable authentication method. However, the widespread use of biometric identification systems, inevitably poses a threat to personal privacy. Therefore, biometric template protection schemes are proposed to cope with the security and privacy problems of biometrics [5, 16, 20, 37, 48, 52, 53, 55, 56, 69, 71, 99]. These schemes are analyzed in detail in deliverables D5.1: Privacy Preservation Techniques and implemented in D5.2: Reference Privacy Preservation. However, various vulnerabilities of these technologies are reported in the literature [1, 2, 13, 21, 22, 45, 46, 57, 60, 61, 63, 80, 83, 90, 93, 94, 96].

Apart from template protection schemes, the employment of cryptographic primitives (i.e. encryption, hashing) offer important advantages in order to safeguard and guarantee the privacy preservation of biometric templates. On the other hand, state-of-the-art cryptographic protocols are not designed for error-tolerance in their inputs, whereas biometric systems have to employ a classifier which tolerates some amount of fuzziness in its data. Thus, combining biometrics and cryptographic protocols to develop a secure system is a very hard problem. It can be dealt with in two ways: either develop a stable feature from biometric data by using error correction methods or make the matching algorithm a part of the cryptographic protocol. However, both approaches are quite hard, and if the design is not done properly then the solution can be a problem itself.

In this deliverable, we investigate robust privacy-preserving biometric authentication schemes that rely mainly on *secure multi-party computation (SMPC)*. SMPC [40, 100] is often used in the design of systems that allow to maximize the utility of information without compromising the user privacy. It involves several distrusting entities $p_1, p_2, \ldots, p_N$ that jointly compute some public function F based on some individually secret information $d_1, d_2, \ldots, d_N$, without revealing their private inputs to one another. More precisely, in SMPC the interactive computation is performed in such a way that participants learn the correct output (i.e. value of the function) and nothing else (e.g. other members' inputs) even if some of the parties maliciously collude to obtain more information. The most often used basic building blocks or primitives of SMPC include homomorphic encryption, oblivious transfer and garbled circuits.

Since biometric systems typically consist of two parties, namely, a client and a service provider, we mainly focus on secure two-party computations (2PC). We investigate how secure 2PC schemes can be utilised to protect sensitive biometric information and thus to preserve privacy in biometric authentication. Particularly, we describe how oblivious transfer, garbled circuits and homomorphic encryption can be employed to guarantee privacy-preservation in biometric authentication. These cryptographic tools can be understood, at a high level, as follows.

– *Oblivious transfer:* An oblivious transfer is a secure two-party computation protocol enabling one party, the sender $\mathcal{S}$ to send one element out of N of elements, to a receiver $\mathcal{R}$ in such a way that the sender $\mathcal{S}$ does not know which element is received by $\mathcal{R}$. At the same time $\mathcal{R}$ does not know anything about the other elements except for the one that is

received.

– *Garbled circuits:* Garbled circuits were introduced by Yao et al. in 1986 [100] in a secure two party setting. A garbled circuit is an encryption of a binary circuit representing the function to be evaluated. Yao's protocol, which is based on garbled circuits and oblivious transfer, was later proven to be provable secure [65].

– *Homomorphic encryption:* Homomorphic encryption was introduced in 1978 by Rivest, Adleman and Dertouzos [84] to go beyond the storage and retrieval of encrypted data by permitting encrypted data to be operated on for interesting operations, in a public fashion – i.e. without decrypting the ciphertexts. An homomorphic encryption scheme enables certain algebraic operations to be carried out on the encrypted plaintext, by applying an efficient operation to the corresponding ciphertext. More precisely, a number of public key cryptosystems have a property called homomorphism such that after encrypting a message, there is some structure preserved that can be exploited to process the data in its encrypted form. In particular, this means that an operation on the encrypted data corresponds to some useful operation on the plaintexts. The only apparent weakness in the homomorphic cryptosystems is that they are computationally expensive.

In addition, we also discuss *fuzzy extractors*. Loosely speaking, a fuzzy extractor is a randomness extractor that comprises of a pair of randomised procedures called a generation algorithm and a reproduction algorithm. The generation algorithm generates a random string and helper data upon an input. The reproduction algorithm reproduces the same random string using the helper data and an input that is *close* to the original input to the generation algorithm. In this deliverable, we focus on robust and reusable fuzzy extractors that can be employed in connection with biometric authentication systems to protect privacy.

Moreover, we give highlights on the latest privacy-preserving biometric authentication protocols that incorporate these cryptographic primitives to provide security and privacy. Even though these latest schemes provide nice properties and a fair level of security and privacy, they are *not* perfect privacy-preserving biometric authentication protocols. There are adversarial scenarios and attacks that can pose a threat against these protocols. We highlight some of the most serious threats and attacks against existing protocols.

The structure of this deliverable is as follows. In Section 2, we present privacy-preserving biometric systems that are built using SMPC. In particular, how oblivious transfer, garbled circuits and homomorphic encryption can be employed to guarantee privacy-preservation in biometric authentication systems is explained in detail, as well as the latest privacy-preserving biometric authentication protocols. Then, in Section 3, robust and reusable fuzzy extractors are described. Section 4 presents attacks on the distributed biometric authentication protocols discussed in Section 2. Finally, we conclude this deliverable in Section 5.

2 Privacy-Preserving Biometrics Using Secure Multi-Party Computation

2.1 Presentation

We here focus on techniques that guarantee full privacy of biometric data using encryption techniques and a multi-party model to distribute the data and keys, rather than encoding techniques with a storage model.

In the context of SMPC [100], a set of parties engage an interactive computation on their inputs in such a way that no information leaks about the inputs that cannot be deduced from the outputs. SMPC is a hot topic in the cryptographic community. While having mostly been of theoretic interest during the 80s and the 90s, it has recently become increasingly practical. Many generic or specific protocols and implementations have been introduced during the last decade and some of these techniques can be applied to the security of Cloud Computing applications. Such techniques are even already used in real-life applications [12].

In particular, several proposals [6, 10, 18, 19, 34, 44, 75, 88, 92] consider the application of SMPC to biometric identification protocols. In a biometric identification process, a server $\mathcal{S}$ holds a database of biometric templates and a client $\mathcal{C}$ owns one biometric sample. The client $\mathcal{C}$ and the server $\mathcal{S}$ execute an authentication protocol which grants access when a fresh biometric sample is similar, w.r.t. a given metric, to one of the elements (biometric templates) stored in the database. The result of the computation can be, for instance, a boolean indicating if there is a match in the database, or the index of the closest element in the database (close set), or a list of indices of the closest elements, or a probabilistic measure of similarity or dissimilarity between templates.

We thus focus on the secure 2PC setting, where the parties are a server $\mathcal{S}$ and a client $\mathcal{C}$. These parties can interactively compute biometric identification without revealing to each other the biometric data that are involved in the protocol; something quite important given the privacy concerns of biometrics. Many applications could benefit from such security properties, such as anonymous biometric access control or private biometric database intersection (see Section 2.1.1).

Security and Privacy. Using a SMPC protocol to compute a given functionality guarantees [64]:
– Correctness: Each party is guaranteed that the output it receives is correct;
– Privacy: Each party should not learn any further information than the prescribed output.

In our context, this means that only an identification result is given as output and that the client $\mathcal{C}$ never gives his biometric acquisition in the clear to the server $\mathcal{S}$ (while the server $\mathcal{S}$ does not disclose his database to the client $\mathcal{C}$). The privacy notion goes even further: if only a match/non-match answer is given as output, no information about the actual score or the level of similarity of biometric data is disclosed.

Security is guaranteed in a specific model of adversaries. We here focus on the case of *semi-honest* (or *passive*) adversaries, that follow the protocol but try to learn more

information than they should. This is currently the only setting where SMPC protocols can be efficiently implemented.

2.1.1 Use Cases

To motivate the use of secure 2PC for biometric identification, we give a non exhaustive list of use cases to which this setting, described in Section 2.2, applies. In some of the cases, the requirements for client's privacy can be for instance enforced by law, especially if it is not in the interest of the server.

Anonymous Biometric Access Control. Assume that there is a biometric access control, for instance, to a company building. The employer $\mathcal{S}$ wants to ensure that the registered (and biometrically enrolled) employees are the only ones to enter the building. To prevent the employer from tracking his employees' activities, our setting is adapted. We can choose the output of the computation to be only a yes/no answer, informing $\mathcal{S}$ whether $\mathcal{C}$ is one of his employees or not, thus allowing/forbidding him entrance to the building.

Biometric Anonymous Credentials. This example, close to the previous one, involves three parties: a client $\mathcal{C}$, a service provider $\mathcal{SP}$ and a biometric server holder $\mathcal{S}$. $\mathcal{S}$ can be for instance a government or an agency authorized to store biometric data, while $\mathcal{SP}$ is not allowed to do so. Let us assume that $\mathcal{S}$ holds a database of people satisfying a given criterion (e.g. be over 21), that is a requirement for the services of $\mathcal{SP}$. To access the services of $\mathcal{SP}$, $\mathcal{C}$ identifies against the database of $\mathcal{S}$. If $\mathcal{C}$ matches the database, $\mathcal{S}$ gives him a token to present $\mathcal{SP}$ to prove that he fulfils the requirements. In the whole process, $\mathcal{C}$ never reveals his identity to other parties.

Secure Biometric Database Intersection. Let us assume that two security agencies want to find out the suspects they have in common or that the police wants to find out which people registered in a given database belong to a list of suspects. For security reasons or due to some privacy policy, the parties involved might want to keep secret the data that are not common to both of them. Our setting described in Section 2.2 can be adapted to this case, by letting the client also input a list of biometric data.

Organization of this section. In Section 2.2, we describe the setting of privacy-preserving biometric identification using SMPC techniques. In the following sections, we describe the main SMPC tools that are employed in this context: oblivious transfer in Section 2.3, garbled circuits in Section 2.4, GMW protocol in Section 2.5 and homomorphic encryption in Section 2.6. We give some elements of comparison between these tools in Section 2.7. We describe the proposals of the state-of-the-art using these tools in Section 2.8, where applications to iris, face and fingerprint are exposed. Finally in Section 2.9, we list the latest distributed privacy-preserving biometric authentication protocols.

2.2 Setting

We here consider the case of biometric (1-vs-N) identification where biometric data can be represented as fixed-length vectors in a metric space and where biometric matching is

done by computing distances between biometric data and comparing these distances to a threshold. Our setting is formalized in Figure 1. The server $\mathcal{S}$ inputs a list of biometric templates $x^1, \ldots, x^N \in \{0,1\}^n$ while the client $\mathcal{C}$ inputs one fresh biometric acquisition $y \in \{0,1\}^n$. All distances $d(x^j, y)$ are computed, for $j = 1, \ldots, N$ and are compared to a threshold t (or to several thresholds $t_1, \ldots, t_N$). Several options for the output are described in Figure 1 depending on the functionality one wants to offer. For the examples given in Section 2.1.1, one possible solution is the output option 5 of Figure 1, where only one bit is given as output.

Inputs

- $\mathcal{S}$: $x^1, \ldots, x^N \in \{0,1\}^n$
- $\mathcal{C}$: $y \in \{0,1\}^n$
- Common inputs:
  - the distance operation $d : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^k$;
  - a threshold $t \in \{0,1\}^k$ or several thresholds $(t_i)_{i\in[1,N]} \in (\{0,1\}^k)^N$ (they can also be private inputs of $\mathcal{S}$).

Outputs

Several output options are possible. One party (usually server $\mathcal{S}$), or both, is given the output.

1. $(d(x^j, y))_{j\in[1,N]}$: the set of distances;
2. $(d(x^j, y) \overset{?}{<} t)_{j\in[1,N]}$: the matching elements in the database;
3. $j^* = \underset{j}{arg\ min}\ (d(x^j, y))$: the closest match;
4. $\begin{cases} j^* = \underset{j}{arg\ min}\ (d(x^j, y)) & \text{if } d(x^{j^*}, y) < t \\ \epsilon & \text{otherwise} \end{cases}$: the closest match, if the distance is below the threshold;
5. $\begin{cases} 1 & \text{if } \exists j : d(x^j, y) < t \\ 0 & \text{otherwise} \end{cases}$: a bit denoting if there is a match or not;

Figure 1: The privacy-preserving biometric identification setting.

2.3 Oblivious Transfers

1-out-of-N oblivious transfer, denoted by OT_1^N, is an elementary secure 2PC protocol enabling one party, the sender $\mathcal{S}$, to send one element out of N bit-strings $(x_0, \ldots, x_{N-1})$

in his possession to a second party, the receiver $\mathcal{R}$. The receiver $\mathcal{R}$ chooses the index $i \in [0, N-1]$ of the element that he would like to get. At the end of the protocol, $\mathcal{R}$ should get x_i, but no information about $(x_j)_{j \neq i}$, while $\mathcal{S}$ should get no information about i. We can write this as:

$$OT_1^N : ((x_0, \ldots, x_{N-1}); i) \mapsto (\epsilon; x_i).$$

In the following, we mostly focus on 1-out-of-2 oblivious transfer:

$$OT_1^2 : ((x_0, x_1); i) \in (\{0,1\}^*)^2 \times \{0,1\} \mapsto (\epsilon; x_i)$$

and parallel execution of n OT_1^2 on λ-bit strings, that we denote by:

$${}_n^{\lambda}OT_1^2 : ((x_0^1, x_1^1), \ldots, (x_0^n, x_1^n)); (i_1, \ldots, i_n)) \in (\{0,1\}^{\lambda})^{2n} \times \{0,1\}^n \mapsto (\epsilon; (x_{i_1}^1, \ldots, x_{i_n}^n)).$$

Oblivious transfer was introduced, in a slightly different form, by Rabin [81], while a protocol computing the actual OT_1^2 functionality first appeared in [36]. Lots of works have focused on possible instantiations of the OT_1^2 and ${}_n^{\lambda}OT_1^2$ protocols and on optimizations to make them efficient. The Naor-Pinkas protocol [72], secure in the random oracle model, is often used in implementations. It relies on the decisional Diffie-Hellman assumption and can be implemented either on sub-groups of prime order of the unit group of a large prime field, or on elliptic curves. Most ${}_n^{\lambda}OT_1^2$ protocols, as [72], require to compute one or several public-key operations that are computationally expensive. Therefore, Ishai *et al.* [47] proposed an extension that makes the number of public-key operations independent of n, for large enough n. The cost is reduced to k "real" oblivious transfers on k-bit values (where k is a security parameter, equal, e.g., to 80 or 128), while there are $\mathcal{O}(n)$ evaluations of cryptographic hash functions, whose cost is substantially cheaper than the cost of public-key operations, this optimization still requires a communication of $2n(\lambda + k)$ bits. Another extension, due to Beaver [8], enables to delegate most of the computational workload of oblivious transfers to a pre-computation phase[1]. It roughly consists of running OTs on random inputs and using the results as masks for the online phase. The *online* phase, then, mostly consists of sending data ($2n(\lambda + 1)$ bits) from the sender $\mathcal{S}$ to the receiver $\mathcal{R}$ for a negligible computational cost (XOR evaluations) on both sides.

2.4 Garbled Circuits

In his seminal paper on Secure Multi-Party Computations [100], Yao describes a protocol that enables to securely compute (in the semi-honest 2PC setting) any function that can be expressed as a binary circuit. Yao's protocol has been later more formalized and proven secure in [65, 66]. This protocol is based on oblivious transfer and *garbled circuits*. Let us describe the latter technique.

A *garbled circuit* (GC) is an encryption of a binary circuit C_f representing the function f to be evaluated. We assume that all gates of the binary circuit have two input wires and one output wire, which is non restrictive.

[1]The preprocessing phase is sometimes called the *offline* phase in SMPC papers. However, this pre-computation phase requires communication between the parties and is not off-line in a strict sense.

Garbled Values. To each wire u of the circuit, we associate a pair of keys $(k_i^0, k_i^1) \in (\{0,1\}^k)^2$. Each of the keys corresponds to a possible binary value of the wire (0 or 1). Computation will be done on these keys, instead of actual bits, therefore we have to redefine binary gates of the circuit.

Garbled Tables. Let G be a gate of C_f with input wires u, v and output wire w. For example, let G be an AND gate. We see G as a truth table and translate this truth table to a truth table with garbled values, replacing bit values by associated keys. For example, with an AND gate:

u	v	w
0	0	0
0	1	0
1	0	0
1	1	1

$\Longrightarrow$

u	v	w
k_u^0	k_v^0	k_w^0
k_u^0	k_v^1	k_w^0
k_u^1	k_v^0	k_w^0
k_u^1	k_v^1	k_w^1

For Yao's protocol, we need to go one step further. The output wire keys are encrypted using the corresponding input keys, then the rows are permuted. In our example, a possible final garbled table is:

$E^s_{k_u^1,k_v^1}(k_w^1)$
$E^s_{k_u^1,k_v^0}(k_w^0)$
$E^s_{k_u^0,k_v^0}(k_w^0)$
$E^s_{k_u^0,k_v^1}(k_w^0)$

where $E_{k_1,k_2}(k_3)$ is an encryption scheme taking as inputs 2 keys k_1, k_2, a message k_3 and an additional information s (that is a unique identifier for gate G).

This way, one can decrypt a row of the garbled gate if one has one key per input wire. Conversely, if one has one key only per input wire, one gets one key only for the output wire. Furthermore, the permutation of the rows prevents the garbled gate evaluator to know the actual bits corresponding to the keys that are manipulated. A garbled circuit is then a set of garbled tables, one per binary gate of the circuit, constructed using the same set of garbled values.

Yao's Protocol. Yao's protocol implements the functionality $(x; y) \mapsto (f(x, y); f(x, y))$ for any deterministic function f that can be described using a binary circuit C_f. Party P_1 acts as the *creator* of the garbled circuit and party P_2 as the *evaluator*. The protocol runs as follows.

1. *Creating and sending the garbled circuit.* Party P_1 selects a set of garbled values for all the wires of the circuit C_f and then computes the garbled tables that are compatible with these garbled values, for all binary gates of the circuit. The resulting set of garbled tables is a garbled circuit $\tilde{C}_f$. Then P_1 sends $\tilde{C}_f$ to P_2.

   Furthermore, P_1 needs to send *output mapping tables* for the output wire keys that will enable P_2 to know the actual bits of $f(x, y)$ that are associated with the output keys that he will output at the end of the evaluation.

2. *Sending garbled values.* In order to allow party P_2 to evaluate garbled circuit $\tilde{C}_f$, P_1 needs to send P_2 one key per input wire of the circuit. These keys should be the keys corresponding to the actual bits of x and y. We denote by $(u_i)_{i=1,\ldots,n}$ the wires corresponding to the bits of x and by $(u_j)_{j=1,\ldots,n}$ the wires corresponding to the bits of y. The party P_1 knows x and can thus directly send to P_2 the keys $k_{u_i}^{x_i}$, for $i = 1, \ldots, n$.

   Sending the keys associated with P_2's inputs is trickier. In order to preserve privacy, P_2 can not give his input y to $P1$. Conversely, P_1 cannot give both keys for each input wire, which would enable P_2 to evaluate $\tilde{C}_f$ on several inputs, and thus learn more information about x than he should. Consequently, P_1 and P_2 have to run oblivious transfers, more precisely, if n is the bit-size of y and k is the security parameter for the protocol, then P_1 and P_2 have to run a ${}_n^k OT_1^2$ protocol, where P_1 inputs all key pairs for wires u_j and P_2 inputs the bits y_j of y.

3. *Evaluating the garbled circuit.* Now that P_2 has the garbled circuit, one key per input wire and the output mapping table, he is fully able to evaluate $\tilde{C}_f$. Party P_2 thus obtains $f(x, y)$ and sends it to P_1 so that both parties learn the result. (Recall that we are in the semi-honest model and that the parties are assumed to follow the protocol, in particular it means that P_2 provides P_1 with $f(x, y)$ even if this result does not suit him.)

This protocol is summed up in Figure 2.

Remark 1. *A garbled circuit can only be used once. Indeed, if one plays Yao's protocol on the same garbled circuit using different inputs, one could decrypt more garbled table rows than expected and learn information about the other party's input that should not have been learned. Therefore, a new set of garbled values and, thus, a new set of garbled tables have to be used if P_1 and P_2 wish to securely evaluate f again, on different inputs.*

Optimizations. During the last decade, several optimizations have been proposed to make Yao's protocol practical. Moreover, the OT optimizations described above can of course be applied to an implementation of Yao's protocol. For more details about these optimizations, we refer the reader to [43, 79, 91].

An efficient encryption construction for the garbled tables components has been proposed in [67, 79], based on cryptographic hash functions. Encryption can roughly be instantiated as follows:

$$E^s_{k_1,k_2}(k_3) = k_3 \oplus \mathcal{H}(k_1||k_2||s),$$

where s is a unique identifier for the gate and $\mathcal{H}$ is a cryptographic hash function. This construction is secure in the random oracle model.

The *point-and-permute* technique [73] enables the evaluator of the garbled circuit to know which entry of the garbled table he has to decrypt, without impacting privacy, thus reducing the evaluator's workload to one hash function evaluation per garbled gate. To do so, each garbled value includes one bit of permutation, in addition to the wire key.

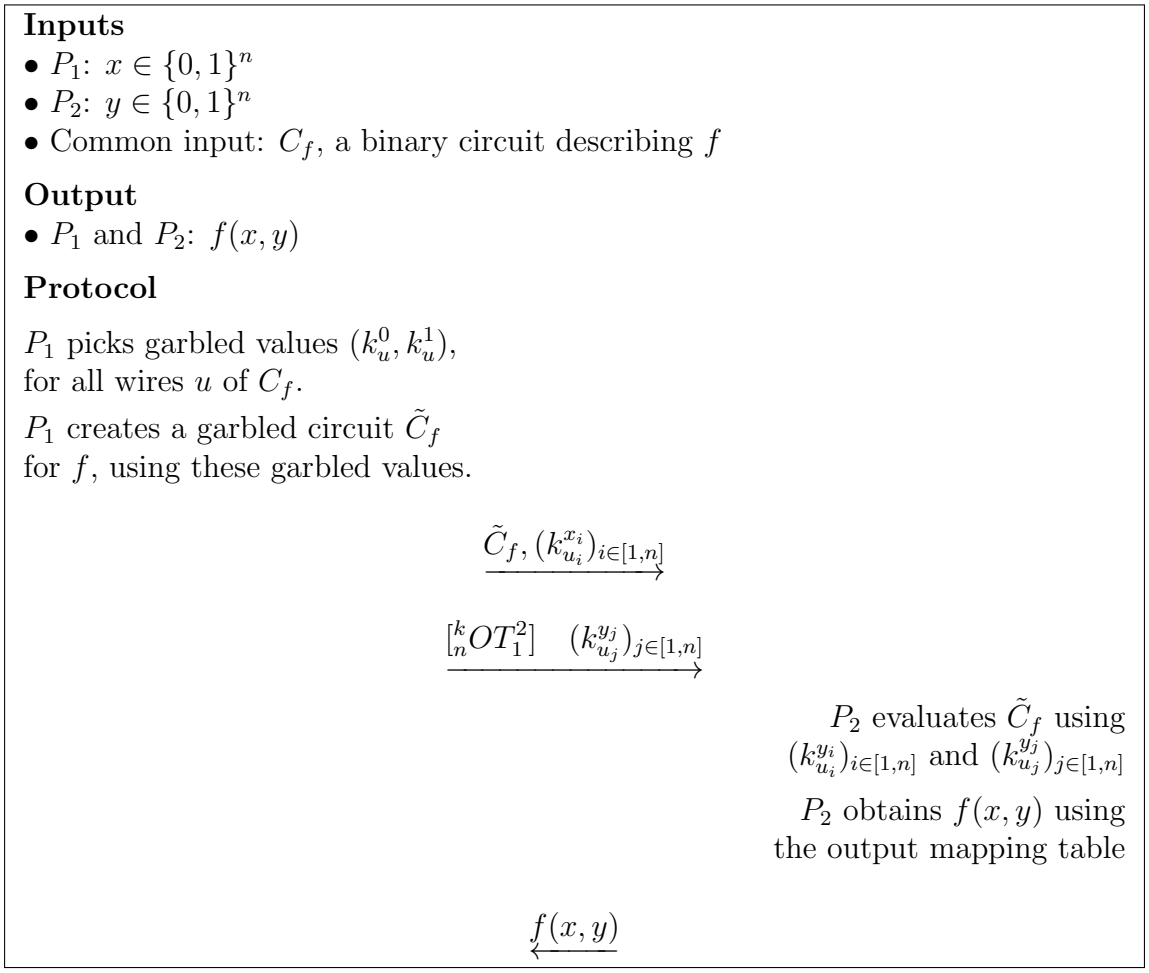

Figure 2: Yao's protocol

The *Free XOR* optimization [59] enables to garble XOR gates "for free". Using this technique, the XOR gates of the original circuits are not garbled, which means that the time spent on encrypting, sending and evaluating these gates can be avoided. It is accomplished by fixing a specific difference between the two garbled values of each garbled wire, *i.e.* for a given garbled circuit, we fix $\Delta \in \{0,1\}^k$ and enforce $k_u^1 = k_u^0 \oplus \Delta$, for each wire u. (Notice that Δ should not be learned by the evaluator.) Moreover, the garbled values of the output wire w of a XOR gate whose input wires are u and v are not randomly picked but computed this way:

$$k_w^0 = k_u^0 \oplus k_v^0 \quad ; \quad k_w^1 = k_w^0 \oplus \Delta.$$

One can then easily be convinced that, for any pair of bits $(a,b) \in \{0,1\}^2$, $k_w^{a\oplus b} = k_u^a \oplus k_v^b$, which is precisely what the evaluator of the garbled circuit will compute instead of evaluating a "real" garbled gate. This optimization is very important and has an impact on the circuit representation. When designing a circuit for integration in Yao's protocol, one should not optimize the total number of gates of the circuit but the number of non-XOR

gates, even by adding more XOR gates, since their impact on the overall execution time is negligible. We refer the reader to [58, 59] for a description of circuits minimizing the number of non-XOR gates for basic functionalities (addition, subtraction, comparison, multiplexer, minimum, multiplication, ...). For example, addition, comparison or subtraction on n-bit inputs requires n non-free XOR gates only.

The *Garbled Row Reduction* technique [72, 79] enables to reduce the size of a garbled table to 3 elements[2], instead of 4, by appropriately picking the garbled values, see [79, Section 4] for more details. For instance, for a security parameter equal to 80 (resp. 128), a garbled non-XOR gate is 240-bit (resp. 384) long. This optimization substantially reduces bandwidth consumption, which is often a bottleneck when deploying 2PC protocols. However, the workload of the creator does not change (4 encryptions).

Finally, the *Pipelined Circuit Execution* technique [43] aims at reducing the computation time during an execution of Yao's protocol. When using pipelined execution, the creator does not wait for encrypting the whole circuit before sending it. Reciprocally, the evaluator does not wait for getting the whole garbled circuit before starting to evaluate it. This does not save communication complexity but it reduces overall computation time and can save memory space on both parties' devices.

2.5 GMW Protocol

GMW protocol [40] was introduced shortly after Yao's protocol. It is not specific to 2PC but can be used with any number of parties. We here sum up the 2PC version. This protocol also considers a function f expressed as a binary circuit. Parties P_1 and P_2 share their input bits between both of them. For example, if x_i is an input bit of P_1, then P_1 picks a random bit r_i, sends r_i to P_2 and holds $x_i \oplus r_i$ as his secret share of x_i. Once all inputs are shared, the circuit is evaluated on the shares. When evaluating a gate, each party owns a share of both inputs and learns a share of the output. Let us respectively denote by (u, v) and w the inputs and output of a gate G. Party P_1 holds shares u_1 and v_1 of u and v, party P_2 holds shares u_2 and v_2 of u and v (*i.e.* $u = u_1 \oplus u_2$ and $v = v_1 \oplus v_2$); they interactively obtain shares of w as follows:

- *XOR gates.* Each party P_i locally computes $w_i = u_i \oplus v_i$.
- *AND gates.* Parties P_1 and P_2 run an OT_1^4 where, say, P_1 is the sender and P_2 is the receiver. P_1 picks a random bit w_1 as his share of $w = u \wedge v$. Party P_1 pre-computes all possible shares of w, depending on the values of u_2 and v_2. More precisely, the OT input at position $(a, b) \in \{0, 1\}^2$ is $w_2 = w_1 \oplus ((u_1 \oplus a) \wedge (v_1 \oplus b))$. Thus, on input (u_2, v_2), P_2 obtains $w_2 = w_1 \oplus ((u_1 \oplus u_2) \wedge (v_1 \oplus v_2)) = w_1 \oplus w_2$, which ensures the correctness.

At the end of the evaluation, P_1 and P_2 exchange their shares of the output bits, and thus obtain the output $f(x, y)$.

[2]This number can even be reduced to 2, if one does not want to use the free XOR technique.

Recently, GMW protocol has been implemented with new optimizations that help make it more efficient (see [3, 92] for more details). The performances in the case of secure evaluation of distances or of biometric identification computation are given in Section 2.8.

2.6 Homomorphic Encryption

Homomorphic cryptosystems [85] are cryptosystems that enable to "compute over encrypted data". More precisely, a cryptosystem E is homomorphic for operation $\boxplus$ (on plaintexts) if there exists an efficiently computable operation $\boxtimes$ (on ciphertexts) such that one can compute an encryption of $m_1 \boxplus m_2$ from encryptions of m_1 and m_2 as

$$E(m_1 \boxplus m_2) = E(m_1) \boxtimes E(m_2),$$

without the knowledge of the secret key. Textbook RSA [86] and ElGamal [38] cryptosystems are homomorphic for the multiplication operation on plaintexts, they are called *multiplicatively homomorphic*. Constructing a cryptosystem that is homomorphic for any operation on plaintexts (called *fully homomorphic*) has been a challenge for a long time, that has only been solved in 2009 by Gentry [39]. Unfortunately, these schemes are for now rather inefficient and cannot be deployed for practical applications.

If the homomorphic properties of the cryptosystem enable to compute $E_{pk}(f(x, y))$ from $E_{pk}(x)$ and y, without knowing the secret key, then one can design a simple 2PC protocol for securely evaluating f, as described in Figure 3.

In the following, we focus on *additively homomorphic* schemes, *i.e.* schemes that are homomorphic for addition on plaintexts. We denote, respectively, by pk, sk, E_{pk}, D_{sk} the public key, secret key, encryption algorithm and decryption algorithm of such schemes. The homomorphic operation on ciphertexts corresponding to additions is denoted multiplicatively, *i.e.* $E_{pk}(m_1+m2) = E_{pk}(m_1) \cdot E_{pk}(m_2)$, for any messages m_1, m_2. Notice that, by induction, one can also obtain $E_{pk}(n \cdot m)$, from $E_{pk}(m)$ and $n \in \mathbb{Z}$, as $E_{pk}(n \cdot m) = (E_{pk}(m))^n$. This means, if we see n as an input to the computation, that one can compute multiplications with an additively homomorphic cryptosystem, without interaction, as long as one of the operands is in the clear. This will be useful for secure biometric matching. Some variants might be applied to the generic protocol described in Figure 3, in order to broaden the set of functions that can be securely evaluated using additively homomorphic encryption.

Interactively computing multiplications. If at some point of the protocol, party P_2 owns two ciphertexts $E_{pk}(u)$ and $E_{pk}(v)$ (while P_1 holds decryption key sk) and needs to compute $E_{pk}(u \cdot v)$, then P_1 and P_2 can proceed as follows:

1. P_1 picks two random numbers r and r', according to the uniform distribution over the plaintext space, and computes $c_1 = E_{pk}(u + r)$ and $c_2 = E_{pk}(v + r')$ using homomorphic properties of E.

2. P_2 sends c_1 and c_2 to P_1.

Inputs
• P_1: $x \in \{0,1\}^n$
• P_2: $y \in \{0,1\}^n$
• Common inputs: a homomorphic cryptosystem E ; a function f, such that $(E_{pk}(u), v) \mapsto E_{pk}(f(u,v))$ can be computed without the knowledge of sk, for any $u, v \in \{0,1\}^n$.

Output
• P_1 and P_2: $f(x,y)$

Protocol

P_1 computes a key pair (sk, pk) for E.
P_1 computes $c = E_{pk}(x)$.

$\xrightarrow{c, pk}$

P_2 computes $c' = E_{pk}(f(x,y))$ using homomorphic properties.

$\xleftarrow{c'}$

P_1 decrypts c' and obtains $D_{sk}(c') = f(x,y)$

$\xrightarrow{f(x,y)}$

Figure 3: Secure 2PC using homomorphic encryption

3. P_1 decrypts $u' = D_{sk}(c_1)$ and $v' = D_{sk}(c_2)$, then computes $c_3 = E_{pk}(u' \cdot v')$.
4. P_1 sends c_3 to P_2.
5. P_2 computes $c_4 = (E_{pk}(u))^{-r'} = E_{pk}(-u \cdot r')$, $c_5 = (E_{pk}(v))^{-r} = E_{pk}(-r.v)$ and $c_6 = E_{pk}(-r \cdot r')$.
6. P_2 computes $c_7 = c_3 \cdot c_4 \cdot c_5 \cdot c_6$, which is an encryption of $u \cdot v$.

The last line is justified using homomorphic properties of E:

$$\begin{aligned} c_3 \cdot c_4 \cdot c_5 \cdot c_6 &= E_{pk}((u+r)\cdot(v+r')) \cdot E_{pk}(-u \cdot r') \cdot E_{pk}(-v \cdot r) \cdot E_{pk}(-r \cdot r') \\ &= E_{pk}((u+r)\cdot(v+r') - u \cdot r' - v \cdot r - r \cdot r') \\ &= E_{pk}(u \cdot v). \end{aligned}$$

Packing. It can be useful to encrypt several messages in the same ciphertext, in order to save bandwidth when ciphertexts are sent from one party to the other. Indeed, the plaintext space is often very large compared to the actual size of data on which we would

like to compute (see description of Paillier encryption below). The idea to deal with this problem is to *pack* several messages into one ciphertext. Let $m_1, \ldots, m_k \in \{0,1\}^\ell$ be such messages, such that $k \times \ell \leq N$, where N is the bit-size of plaintexts of E. Then one can encrypt $m_1, \ldots, m_k$ in the same ciphertext by setting the plaintext to $m = \sum_{i=1}^{k} m_i 2^{\ell \cdot (i-1)}$, which is equivalent to encrypting $m_k || \cdots || m_1$, where $||$ denotes concatenation. Packing enables to batch homomorphic additions. (Unfortunately, one cannot batch homomorphic multiplication with several operands in the clear).

Re-randomizing ciphertexts. It can be useful for privacy to re-randomize ciphertexts. (Notice that homomorphic schemes used for secure 2PC have a randomized encryption algorithm, in order to satisfy ciphertext-indistinguishability requirements, see below.) Given a ciphertext $c = E_{pk}(m)$, it suffices to compute $c' = c \cdot E_{pk}(0)$, which is also an encryption of m, thanks to the additively homomorphic property of E.

Two main additively homomorphic schemes are usually employed in implementations of secure biometric identification: Paillier cryptosystem [77] and DGK cryptosystem [24–26]. We give some details on these schemes below. Notice that lifted ElGamal [38] can also be used if the messages that are decrypted lie in a small set, since decryption ends with a search for a discrete logarithm.

Paillier cryptosystem. Paillier cryptosystem [77] is an additively homomorphic cryptosystem, whose security is based on the decisional composite residuosity problem. The public key consists of a κ-bit RSA number $N = pq$, where p and q are prime, and $g \in \mathbb{Z}_{N^2}$ of order a multiple of N in $\mathbb{Z}_{N^2}^*$, for instance $g = N+1$ [27]. The private key consists of p and q. The plaintext space is $\mathbb{Z}_N$ and the ciphertext space is $\mathbb{Z}_{N^2}$. To encrypt a message $m \in \mathbb{Z}_N$, one selects a random number $r \xleftarrow{\$} \mathbb{Z}_N$ and computes a ciphertext $c = E_{pk}(m) = g^m \cdot r^N \in \mathbb{Z}_{N^2}$.

As for RSA encryption, one should use 1024-bit N moduli for 80-bit security and 3072-bit moduli for 128-bit security. Notice that the ciphertext space is $\mathbb{Z}_{N^2}$, thus the ciphertexts have twice the size of the plaintexts. In particular, even the encryption of one bit requires at least 2048 bits for a reasonable level of security, which motivates the use of another scheme when manipulating small plaintexts.

DGK cryptosystem. DGK encryption [24–26] answers to this problem. The public key still includes a κ-bit RSA modulus $N = pq$, in addition to a small integer u, and two elements $g, h \in \mathbb{Z}_N^*$ (with restrictions about their order, linked to u, p and q). Plaintext space is now $\mathbb{Z}_u$, while ciphertext space is $\mathbb{Z}_N$ (instead of $\mathbb{Z}_{N^2}$ for Paillier encryption). Encryption of a message $m \in \mathbb{Z}_u$ is performed by picking a random integer r and computing $c = E_{pk}(m) = g^m \cdot h^r \in \mathbb{Z}_N$.

2.7 Elements of Comparison

There is no absolute best solution for securely evaluating a given function f. It can depend for instance on the computational capabilities of the parties, the latency of the network. We can however give some elements of comparison.

Yao's protocol is more dedicated to functions that have an efficient representation as a binary circuit. It enjoys many optimizations that reduce its cost to few public-key operations and mostly symmetric operations. Moreover, it is a 1-round protocol, which can be useful when suffering from low network latency. The drawbacks are that a garbled circuit can only be used once and that sending garbled circuit and garbled values leads to significant communication costs.

GMW protocol is also more dedicated to binary functions and enjoys optimizations on oblivious transfer. Moreover, compared to Yao's protocol, the cost of sending garbled circuit is removed. However, the number of rounds depends on the depth (not taking XORs into consideration) of the binary circuits.

The solution employing homomorphic encryption is more suited for arithmetic functions on integer inputs. Homomorphic ciphertexts also have the advantage that they can be used for several secure evaluations. The communication cost depends on the size of the inputs and on the multiplicative depth of the function. In particular, encrypting small values with homomorphic ciphertexts can lead to large communication overheads. The number of rounds also depends on the multiplicative depth. The major drawback is that encryption, decryption and homomorphic operations are arithmetic operations (products, exponentiations) on large moduli, which could lead to a significant computational cost.

It is possible to design hybrid protocols that use several tools for different parts of a secure computation. Notice that one should pay attention to the way tools are sequentially combined. In particular, intermediary results should not be output in the clear. Often, a masked output is obtained by one party while the other party holds the mask, which prevents either party from learning information about intermediary outputs.

2.8 State of the Art

In this section, we describe existing protocols for privacy-preserving biometric identification, using 2PC techniques. They make use of the techniques exposed in previous sections using different combinations. We report implementation results exposed in the associated publications.

2.8.1 Fingercodes

The FingerCode representation of fingerprints was introduced by Jain *et al.* in [49]. Although fingerprint representations are usually based on local reference points called minutiae, this representation does not take them into consideration in order to achieve a much simpler comparison algorithm. In a few words, in order to encode the image, one first locates a central reference point and, preferably, a direction in order to define a disk of analysis. Each sector of the disk is filtered using a bank of Gabor filters. For each sector, one computes the standard deviation of the Gabor phases to obtain the FingerCode. The matching operation between two FingerCodes is an Euclidean distance. In proposals described below, experiments were conducted on FingerCodes made of vectors of 16 coordinates, each coordinate being a 7-bit integer.

Using Paillier and ElGamal cryptosystems. Barni *et al.* [6] describe a scheme based on homomorphic encryption (Paillier [77] and additive ElGamal [38] over elliptic curves). Euclidean distance comparison follows the methodology of Figure 3 and uses Paillier encryption. Once server $\mathcal{S}$ holds all encrypted distances between client $\mathcal{C}$'s input and every database element, they run a comparison protocol based on homomorphic encryption, adapting the DGK method [24–26] to the EC-ElGamal cryptosystem [38], in order to save bandwidth.

On a database of 900 individuals, with 5 fingercodes for each identity, *i.e.* $N = 4500$, the overall bandwidth usage is 10 MB for a security level $k = 80$ and 21 MB for a security level $k = 128$. As for timings, performances were evaluated on a database of 64 individuals, with 4 fingercodes each, thus reaching $N = 320$. For a security level $k = 80$, time needed to perform an identification is about 16 seconds on a single PC (*i.e.* not including communication time) and after preprocessing.

Using DGK cryptosystem and Yao's Protocol. Blanton and Gasti [10] described a proposal that uses DGK encryption [24–26] for euclidean distance computation and Yao's protocol for comparison of the distances to a threshold. Performances of the [10] protocol are summed up in the following table:

	Pre-processing	Online phase
Server	1451.6 + 4.3 N ms	0.22 + 1.42 N ms
Client	1086+0.15 N ms	4.7 + 1.08N ms
Communication	11.6 + 1.26 N KB	2.12 + 0.86 N KB

In another proposal, Huang *et al.* [44] also propose a hybrid protocol. It uses Paillier encryption for Euclidean distance computation, with an optimization using packing techniques. Yao's protocol, and a spectific *backtracking* protocol are used for retrieving the index of the closest match. Their implementation considers 640 $\times$8-bit vectors.

Comparison. For a comparison between the above schemes, for $N = 4500$, Barni *et al.* 's solution requires 10 MB of communication, while Blanton and Gasti's requires 3.8 MB of online communication (9.3MB overall).

For $N = 320$, the first solution requires 16 seconds online time, while the second solution's online phase requires 455 ms on the server's side and 350 ms on client's side (resp. 3.3 s and 1.5 s overall). With close parameters, Huang *et al.* claim to take 3.47s computation time and 3.76 MB communication.

2.8.2 Eigenfaces

In 1991, Turk and Pentland introduced a new approach to human face recognition known as *Eigenfaces* [97]. Using this representation, face images in a high-dimensional vector space are transformed into feature vectors in a low-dimensional vector space whose basis is composed of *eigenfaces*. Such a basis is obtained through Principal Component Analysis from a set of training images. We do not detail this process here and only focus on the projection and matching algorithms.

We assume that face images are represented by vectors of length n, and that a set of eigenfaces $u_1, \ldots, u_K$ forming a basis for the low dimensional space has already been set up. Notice that $K < n$. We also need the vector Ψ representing the average of the training images used to compute the u_i's.

Let X be a new face image. We obtain its feature vector representation as follows:

1. Compute $\Phi = X - \Psi = (\Phi_1, \ldots, \Phi_n)$;

2. For each $i = 1, \ldots, K$, compute $\omega_i = \Phi_1.(u_i)_1 + \cdots + \Phi_n.(u_i)_n$;

3. The feature vector associated to X is $\Omega = (\omega_1, \ldots, \omega_K)$.

The distance between two feature vectors Ω and Ω' can then be defined (in the simplified version that we consider) as the squared euclidean distance $D(\Omega, \Omega') = ||\Omega - \Omega'||^2 = (\omega_1 - \omega'_1)^2 + \cdots + (\omega_K - \omega'_K)^2$.

In the secure versions of this protocol, in addition to the feature vectors in the database, the eigenfaces $u_1, \ldots, u_K$ and the average vector Ψ are also assumed to be private inputs of $\mathcal{S}$. Implementations use 192×112-pixel images as inputs, where pixels are 8-bit values, and $K = 12$ eigenfaces. The distances are 50-bit long.

Using homomorphic encryption only. Erkin *et al.* [34] proposed to follow the homomorphic encryption approach. Client $\mathcal{C}$ sends a ciphertext per coordinate of his input image. Then, projection can be homomorphically computed by $\mathcal{S}$ without interactions. However, one round of interaction is required for distance computation since a squaring operation has to be homomorphically performed (contrary to the case of Fingercodes where $\mathcal{C}$ can directly send the sum of the squared coordinates of his input). Projection and distance use Paillier encryption while comparison is done using DGK encryption and DGK comparison protocol [24–26].

Using hybrid solutions. Hanecka *et al.* [42] suggest to still use (Paillier) homomorphic encryption for the projection and distance phases, but to use Yao's protocol for comparison/minimum phases. Their implementation uses the generic TASTY tool [42] and consequently does not contain all possible optimizations. Sadeghi *et al.* [88] propose a more dedicated and thus more efficient implementation using the same hybrid approach.

Using GMW Protocol. Schneider and Zohner [92] implemented a GMW version of the Eigenfaces recognition protocol, reaching very good timing performances, especially for the online phase.

Comparison. We sum up in Table 1 the performances of the protocols exposed in [34, 42, 88, 92]. Dashes indicate when information is not explicitly available (this does not mean that value is zero). HE denotes homomorphic encryption and GC denotes garbled circuits. Notice that some experiments were run on a single computer, other on two computers in a LAN setting. We refer the reader to the associated publications for more details.

	$N = 320$				$N = 1000$		
Protocol (Tools)	[34] (HE)	[88] (HE+GC)	[42] (HE+GC)	[92] (GMW)	[88] (HE+GC)	[42] (HE+GC)	[92] (GMW)
Preproc. (sec.)	22	–	38.1	15.7	–	83.4	24
Online (sec.)	18	8.4	41.5	1.1	13	56.2	1.3
Total (sec.)	40	–	76.9	17.7	–	139.6	26.3
Preproc. (MB)	–	–	3.3	–	–	10.2	–
Online (MB)	–	2.8	5.9	–	3.5	6.8	–
Total (MB)	7.3	–	9.2	–	–	17	–

Table 1: Performances of privacy-preserving Eigenfaces identification protocols

2.8.3 SCiFI (Face)

The SciFI project [50, 74, 75] is a project of the University of Haifa where a new face identification system is proposed. The main particularity of this system is that the biometric identification algorithm has been specifically designed for a more efficient usage in secure computation.

In their system, biometric templates are fixed-length vectors. In this representation a face is represented by p *patches*. A *vocabulary* of M *words* is processed for each patch. In the vectorial representation, there is an appearance component and a spatial component. The spatial component corresponds to the distance between each patch and the center of the face image, it is represented by 2 out of 10 possible distances for each patch. The appearance component is, for each patch, the set of m words, out of the M available in the vocabulary, that are the closest to the real patch of the face image.

The template is a $p.(M + 10)$-bit feature vector. For each patch, 10 bits are used for the spatial component and M for the appearance component. Among the first ten bits, the two bits that correspond to the indices representing the distance between the image and the center of the image are set to 1 and the others are set to 0. Identically for the M other bits, the m bits whose indices correspond to the appearance component are set to 1 and the others are set to 0.

For instance, with $m = 2$ and $M = 10$, if the distance between a given patch and the center of the image is between 6 and 7, and if the closest words in the vocabulary are indexed by 2 and 9, the part of the feature vector corresponding to this patch will be:

	Spatial component										Appearance component									
Bit position	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
Feature vector	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0

Consequently, the feature vectors have a constant Hamming weight equal to $p.(2 + m)$. In the implementation of [75], $p = 30$, $M = 20$ and $m = 4$, this results to 900-bit vectors, in which 180 bits are set to 1. The matching operation is an exclusive-OR between two feature vectors. Experiments described in [75] show that ange of distances is $[0, 180]$.
Using Homomorphic encryption. Osadchy *et al.* [75] propose an implementation of the SCiFI identification protocol that employs homomorphic encryption for distance compu-

	$N = 100$			$N = 320$		$N = 50,000$
Protocol (Tools)	[75] (HE)	[43] (GC)	[92] (GMW)	[43] (GC)	[92] (GMW)	[92] (GMW)
Preproc. (sec.)	213	0.4	0.3	0.4	0.49	44.85
Online (sec.)	31	1.75	0.006	5.14	0.01	0.9
Overall (sec.)	244	8.79	0.31	42.9	0.51	45.98

Table 2: Performances of privacy-preserving SCiFI identification protocols, in a LAN setting

tation and that uses oblivious transfer for comparison evaluation.
Using Yao's Protocol. Huang *et al.*, in their paper [43] aiming at showing the good performances of Yao's protocol (especially by introducing pipelining), describe an implementation of Hamming distance computation using this protocol.
Using SHADE. Bringer *et al.* [18] propose the SHADE protocol that is dedicated to Hamming distance computation. The execution time is approximately divided by 4, compared to Yao's protocol using FastGC.
Using GMW Protocol. Schneider and Zohner [92] show that using GMW protocol leads a very efficient identification protocol that deals in less than 1 second with 50,000 elements in the database.
Comparison. Comparisons of the aforementioned protocols are reported in Table 2. All timings include communication time over a local area network (LAN).

2.8.4 IrisCodes

In the case of iris recognition using IrisCodes [28], biometric templates can be represented as binary vectors. A 256-byte (2048 bits) iris template, together with a 256-byte mask, is computed from an iris image using the algorithm reported in [28]; the mask filters out the unreliable bits, *i.e.* stores the erasures positions of the iris template. The resulting template is called *IrisCode*.

Given an image of the eye, the first step of the encoding algorithm is to find the part of the image that corresponds to the iris area between the pupil-iris and the iris-sclera boundaries. Upon isolating the iris, its texture is normalized using a rubber sheet model in which the iris image is remapped from a Cartesian coordinate system to a polar coordinate system regardless the iris size and the pupil dilation. After normalization, a set of Gabor filters is applied on every direction and location of the normalized and rectangular shaped iris image. Each computed Gabor phase value is then coded into 2 bits depending to its position on the trigonometric circle.

The classical way to compare two IrisCodes relies on a normalized Hamming distance (NH) computation between binary vectors: given $X = (x_1, \ldots, x_{2048})$,$Y = (y_1, \ldots, y_{2048})$ two 2048-bit representations of irises and the associated masks $M(X) = (m_1, \ldots, m_{2048})$

and $M(Y) = (m'_1, \ldots, m'_{2048})$, compute

$$d_{NH}(X,Y) = \frac{||(X \oplus Y) \cap M(X) \cap M(Y)||}{||M(X) \cap M(Y)||} = \frac{\sum_{i=1}^{2048} (x_i \oplus y_i) \cdot m_i \cdot m'_i}{\sum_{i=1}^{2048} m_i \cdot m'_i}, \quad (1)$$

for some rotations of the second template – to deal with the iris orientation's variation – and keep the lowest distance.

Using hybrid solutions. Blanton and Gasti [10] propose to follow the homomorphic encryption approach for normalized Hamming distance evaluation. Comparison operations are done using Yao's protocol. Performances are reported in the following table.

	No rotation		11 rotations	
	Pre-processing	Online phase	Pre-processing	Online phase
Server (ms)	2855 + 7.25 N	89 + 13.71 N	3178 + 79.5 N	89 + 149.25 N
Client (ms)	12,990 + 0.34 N	2.08N	13,620 + 3.39 N	22.61 N
Comm. (KB)	524 + 2 N	0.5 + 1.8 N	524 + 22.1N	0.5 + 19.9 N

Using Yao's Protocol. Bringer *et al.* [19] suggested to use Yao's protocol instead of homomorphic encryption. The aim of Bringer *et al.* was more to introduce a protocol that takes filtering into account than to introduce a faster protocol for secure iris identification. Indeed, they report a performance of about 2.4 seconds per rotation per element in the database.

2.9 Distributed Systems

At a high level, a biometric authentication system consists of a client and a server. However, since the server often has to perform many tasks, his role can be divided to several parts. Therefore, we go further and subdivide a biometric authentication system into components where each component performs a specific task. Thus, we can have a distributed biometric authentication system which consists of five different entities: a user $\mathcal{U}$, a biometric sensor $\mathcal{S}$, an authentication server $\mathcal{AS}$, a database $\mathcal{DB}$ and a matcher $\mathcal{M}$. Such a system works as follows. Let N be the number of users. Then in the enrolment phase, each user $\mathcal{U}_i$, where $1 \leq i \leq N$, submits his/her biometric data b_i to the database $\mathcal{DB}$ to be stored. In the identification (or authentication) phase, the user U_i, $1 \leq i \leq N$ provides a fresh biometric data b'_i and his/her identity ID_i to the sensor $\mathcal{S}$, which in turn forwards these data to the authentication server $\mathcal{AS}$. The $\mathcal{AS}$ then asks the database $\mathcal{DB}$ for U_i's biometric data b_i that is already stored in $\mathcal{DB}$. After getting b_i from $\mathcal{DB}$, $\mathcal{AS}$ sends b_i and b'_i to the matcher $\mathcal{M}$, who checks whether b_i and b'_i matches and sends back the result of the comparison to $\mathcal{AS}$. Finally, $\mathcal{AS}$ outputs to the user U_i either 1 or 0, depending on whether U_i is successfully authenticated or not. See Figure 4 for an illustration of the whole process.

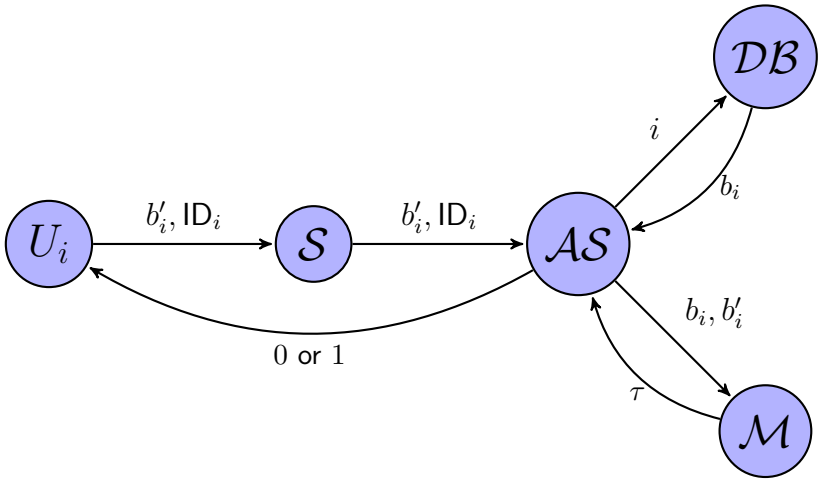

Figure 4: Description of a distributed biometric authentication system.

Recently, some privacy-preserving biometric authentication protocols using, among others, homomorphic encryption have been proposed. Here we present three of them, namely, the protocols by Bringer *et al.* [17], Barbosa *et al.* [4] and Stoianov [95].

2.9.1 The Bringer *et al.* Protocol

In [17], Bringer *et al.* proposed a biometric authentication scheme which follows the above described model using the Goldwasser-Micali cryptosystem [41]. In this protocol the matcher $\mathcal{M}$ generates a pair of public and secret keys (K_p, K_s). The sensor $\mathcal{S}$, the authentication server $\mathcal{AS}$ and the database $\mathcal{DB}$ store the public key K_p while the matcher $\mathcal{M}$ stores the secret key K_s.

Let N be the total number of users in the system. Then the authentication server $\mathcal{AS}$ also stores the mapping (ID_i, i) for $i \in \{1, \ldots, N\}$ where i corresponds to user $\mathcal{U}_i$. In the enrollment step, the database $\mathcal{DB}$ stores the reference biometric templates b_i, for $i \in \{1, \cdots, N\}$, of all users. Let us denote by $\mathsf{Enc}^{\mathsf{GM}}(b_i)$ the *bit-by-bit* encryption of each bit of the template b_i, i.e. $\mathsf{Enc}^{\mathsf{GM}}(b_{i,1} \ldots b_{i,M}) = \big(\mathsf{Enc}^{\mathsf{GM}}(b_{i,1}), \ldots, \mathsf{Enc}^{\mathsf{GM}}(b_{i,M})\big)$, where M is the bit length of the template. The encryption $\mathsf{Enc}^{\mathsf{GM}}(\cdot)$ is such that $\mathsf{Enc}^{\mathsf{GM}}(a)\mathsf{Enc}^{\mathsf{GM}}(b) = \mathsf{Enc}^{\mathsf{GM}}(a \oplus b)$ and $\mathsf{Enc}^{\mathsf{GM}}(c.a) = \mathsf{Enc}^{\mathsf{GM}}(a)^c$. For the sake of simplicity, let us use $\mathsf{Enc}(\cdot)$ to denote $\mathsf{Enc}^{\mathsf{GM}}(\cdot)$ in the following detailed description of the protocol.

The authentication step of the Bringer *et al.* protocol can be divided into the following four phases:

- Phase 1 - Communication $\mathcal{U}_i \rightarrow \mathcal{S} \rightarrow \mathcal{AS}$: In this phase:
  1. $\mathcal{U}_i$ provides a fresh biometric trait b'_i and his identity ID_i to $\mathcal{S}$.
  2. Then, $\mathcal{S}$ sends $\mathsf{Enc}(b'_i)$, the encryption of b'_i the public key K_p, as well as the claimed identity ID_i of $\mathcal{U}_i$ to $\mathcal{AS}$.

- Phase 2 - Communication $\mathcal{AS} \leftrightarrow \mathcal{DB}$: Here:

  1. $\mathcal{AS}$ performs the mapping from ID_i to i and then using a *Private Information Retrieval (PIR)* mechanism sends the identity i of $\mathcal{U}_i$ and requests the corresponding stored biometric template b_i from $\mathcal{DB}$. More precisely, $\mathcal{AS}$ sends to $\mathcal{DB}$ the encrypted value $\mathsf{Enc}(t_j)$, where $1 \leq j \leq N$ and $t_j = 1$ if $j = i$, 0 otherwise.
  2. $\mathcal{DB}$ computes $\mathsf{Enc}(b_{i,k}) = \prod_{j=1}^{N} \mathsf{Enc}(t_j)^{b_{j,k}}$, where $1 \leq k \leq M$, and sends the result to $\mathcal{AS}$.

- Phase 3 - Communication $\mathcal{AS} \leftrightarrow \mathcal{M}$: In this phase:

  1. $\mathcal{AS}$ computes $v_k = \mathsf{Enc}(b'_{i,k})\mathsf{Enc}(b_{i,k}) = \mathsf{Enc}(b'_{i,k} \oplus b_{i,k})$, where $1 \leq k \leq M$. Then, $\mathcal{AS}$ permutes v_k and sends the permuted vector $\lambda_k = v_{\pi(k)}$, $1 \leq k \leq M$, to $\mathcal{M}$.
  2. $\mathcal{M}$ decrypts the permuted vector λ_k and checks whether the Hamming weight (HW) of the decrypted permuted vector is less than a defined threshold τ. The result of this control is sent to $\mathcal{AS}$. (The Hamming weight is equal to the Hamming distance between b_i and b'_i.)

- Phase 4 - Communication $\mathcal{AS} \rightarrow \mathcal{U}_i$: Finally, $\mathcal{AS}$ accepts or rejects the authentication request ($\mathsf{Out}_{\mathcal{AS}} = 1$ or $\mathsf{Out}_{\mathcal{AS}} = 0$ respectively) depending on the value returned by $\mathcal{M}$.

2.9.2 The Barbosa *et al.* Protocol

In [4], Barbosa *et al.* has proposed a protocol that is based on the same biometric authentication model as the Bringer *et al.* protocol composed of the four entities: $\mathcal{S}$, $\mathcal{AS}$, $\mathcal{DB}$ and $\mathcal{M}$. The main differences are that in the Barbosa *et al.* protocol the decisions of the matcher $\mathcal{M}$ are not taken based on the Hamming distance (HD) but based on the Support Vectors Machine (SVM) classifier [23] and that this protocol employs the $\mathsf{Paillier}$ encryption scheme [77, 78].

In this protocol, the sensor $\mathcal{S}$ captures a fresh biometric $b'_i = v = \langle v_1, \ldots, v_M \rangle$, which is represented as a vector of integers. The classifier SVM takes as input N classes (N is the number of users). For each class there are S samples (records). The SVM classifier determines the support vectors $\mathsf{SV}_{i,j}$ and the weights $\alpha_{i,j} \in \mathbb{N}$, where $1 \leq i \leq S$ and $1 \leq j \leq N$, and computes:

$$cl^{(j)}_{\mathsf{SVM}}(v) = v \cdot \left(\sum_{i=1}^{S} \alpha_{i,j} \cdot \mathsf{SV}_{i,j} \right) = \sum_{i=1}^{S} \alpha_{i,j} \sum_{\ell=1}^{M} [v]_\ell \cdot [\mathsf{SV}_{i,j}]_\ell,$$

where $[\cdot]_\ell$ denotes the ℓ-bit of a vector. After computing $cl^{j}_{\mathsf{SVM}}(v)$ the classifier is able to determine the class of the fresh biometric v or to reject it. More precisely, the decision is calculated by finding $\arg\max_j(c^{(j)}_{\mathsf{SVM}}(v))$, where $j = 1, \ldots, N$.

In this protocol $\mathcal{M}$ generates a pair of public and secret keys (K_p, K_s) to be used in the $\mathsf{Paillier}$ encryption scheme. $\mathcal{S}$ stores the public key K_p, while $\mathcal{M}$ stores the secret key K_s. Here the public key K_p defines a modulus $n > 0$ used in the scheme. $\mathcal{DB}$ stores the support vectors $\mathsf{SV}_{i,j}$ and the weights $\alpha_{i,j}$, where $1 \leq i \leq S$ and $1 \leq j \leq M$. We denote by $\mathsf{Enc}^{\mathsf{Paillier}}(v)$ the bit-by-bit encryption of the template v, i.e. $\mathsf{Enc}^{\mathsf{Paillier}}(v_1 \dots v_M) = \big(\mathsf{Enc}^{\mathsf{Paillier}}(v_1), \dots, \mathsf{Enc}^{\mathsf{Paillier}}(v_M)\big)$. For this encryption function $\mathsf{Enc}^{\mathsf{Paillier}}(\cdot)$ it holds that, for $a, b \in \mathbb{Z}_n$, $\mathsf{Enc}^{\mathsf{Paillier}}(a)\mathsf{Enc}^{\mathsf{Paillier}}(b) = \mathsf{Enc}^{\mathsf{Paillier}}(a+b)$ and $\mathsf{Enc}^{\mathsf{Paillier}}(c.a) = \mathsf{Enc}^{\mathsf{Paillier}}(a)^c$, for $c \in \mathbb{Z}$. Again for the sake of simplicity we use $\mathsf{Enc}(\cdot)$ to denote $\mathsf{Enc}^{\mathsf{Paillier}}(\cdot)$ in the following description of the protocol.

The protocol can be divided into the following phases:

- Phase 1 - Communication $\mathcal{U}_i \rightarrow \mathcal{S} \rightarrow \mathcal{AS}$: In this phase the following takes place:
  - $\mathcal{U}_i$ provides a fresh biometric trait $b'_i = v = \langle v_1, \dots, v_M \rangle$ to $\mathcal{S}$.
  - Then, $\mathcal{S}$ sends $w = \mathsf{Enc}(v)$ to $\mathcal{AS}$.
- Phase 2 - Communication $\mathcal{AS} \leftrightarrow \mathcal{DB}$: In this phase:
  - Upon receiving the encrypted fresh biometric w, $\mathcal{AS}$ forwards it to $\mathcal{DB}$.
  - $\mathcal{DB}$ computes $c_j = \prod_{i=1}^{S} \big(\prod_{\ell=1}^{M} [w]_\ell^{[\mathsf{SV}_{i,j}]_\ell}\big)^{\alpha_{i,j}}$, where $j = 1, \dots, N$, which actually corresponds to the encryption of $cl^{(j)}_{\mathsf{SVM}}$. Then, $\mathcal{DB}$ transmits c_j to $\mathcal{AS}$.
- Phase 3 - Communication $\mathcal{AS} \leftrightarrow \mathcal{M}$: This phase involves the following:
  - $\mathcal{AS}$ permutes and re-randomizes c_j (i.e. $\lambda_j = c_{\sigma(j)}$) and transmits the resulting values to $\mathcal{M}$.
  - $\mathcal{M}$ using the private key K_s decrypts λ_j and deduces $cl^{(j)}_{\mathsf{SVM}}(v) = \sum_{i=1}^{S} \alpha_{i,j} \sum_{\ell=1}^{M} [v]_\ell \cdot [\mathsf{SV}_{i,j}]_\ell$, in a permuted form, for all $1 \leq j \leq N$. Finally, $\mathcal{M}$ returns to $\mathcal{AS}$ $j = \arg\max_j cl^{(j)}_{\mathsf{SVM}}$.
- Phase 4 - Communication $\mathcal{AS} \rightarrow \mathcal{U}_i$: Given the value sent by the matcher $\mathcal{M}$ the authentication server $\mathcal{AS}$ determines whether $i = \sigma^{-1}(j)$. If they equal, the user $\mathcal{U}_i$ is authenticated him; otherwise, he is rejected.

2.9.3 The Stoianov Protocol

There are several protocols proposed by Stoianov in [95], but as an example here we select one of them. In this protocol there are three entities: a client $\mathcal{C}$, a service provider $\mathcal{SP}$ and database $\mathcal{DB}$. This protocol employs the Blum-Goldwasser (BG) [11, 70] encryption scheme and the Blum-Blum-Shub pseudo-random generator. In the Blum-Blum-Shub pseudo-random generator, from a seed x_0 and a public key, a pseudo-random sequence S is generated and the state x_{t+1} of the generator is the encrypted value of the seed x_0, i.e. $x_{t+1} = x_0^{2^{t+1}}$, $S_i = \mathsf{lsb}(x_0^{2^i})$, where $i = 1, \dots t$ and t is the length of the plaintext.

In the Stoianov protocol, the $\mathcal{SP}$ generates BG public and private keys (K_p, K_s) and sends K_p to $\mathcal{C}$. In the *enrollment process*, $\mathcal{C}$ computes the BG-encrypted biometric template $(b \oplus S \oplus c, x_{t+1})$, where b is the biometric template captured, S is a pseudo-random sequence of the same length as b generated by the Blum-Blum-Shub pseudo-random generator with a random seed x_0, and x_{t+1} is the encrypted seed. In the *enrollment process*, $\mathcal{C}$ sends $b \oplus S \oplus c$ to $\mathcal{DB}$ for storage, while x_{t+1} and $h(c)$ (hashed codeword) are sent to $\mathcal{SP}$.

The pseudorandom stream S stored in the database $\mathcal{DB}$ and the matcher $\mathcal{SP}$ are periodically updated by employing a stream cipher independent from the BG cryptosystem. At each attempt for biometric authentication the pseudorandom stream will be different denoted as $\check{S}$ and the stored reference template will be $\check{S} \oplus b \oplus c$. It is assumed that the database $\mathcal{DB}$ and the matcher $\mathcal{SP}$ are synchronized. Thus, at each update the pseudorandom stream is also updated at the matcher $\mathcal{SP}$.

In the *authentication process*, $\mathcal{C}$ obtains a fresh biometric template b' and requests $\mathcal{DB}$ to send the enrolled BG-encrypted data $b \oplus \check{S} \oplus c$ by claiming his identity. He generates a new keystream S' using a new seed x'_0 and sends $b' \oplus S' \oplus b \oplus \check{S} \oplus c$ and the new encrypted seed x'_{t+1} to $\mathcal{SP}$. The service provider $\mathcal{SP}$ then using the private key and x'_{t+1} recovers S', and XOR's the received value $b' \oplus S' \oplus b \oplus \check{S} \oplus c$ first with S' and subsequently with the stored $\check{S}$. Thus, $\mathcal{SP}$ recovers $c \oplus b \oplus b'$. Then, $\mathcal{SP}$ runs an ECC decoder algorithm and obtains a codeword c'. If b and b' are close enough, then the original codeword c should be retrieved. Finally, $\mathcal{SP}$ verifies this by checking if $h(c) = h(c')$.

3 Fuzzy Extractors

Basic fuzzy sketches (a.k.a. fuzzy commitment and fuzzy vault) [32, 33, 51, 52, 54] and fuzzy extractors [32, 33] were reviewed in D5.1 as important biometric cryptosystems for template protection and biometric based authentication and key agreement.

Informally speaking, a fuzzy sketch generates some public information P about its input ω such that P by itself does not reveal ω, yet it allows exact recovery of ω given another value ω' which is close to ω. A fuzzy extractor can produce a nearly uniform random value R from its input ω in an error-tolerant manner; that is, even if the input changes to some other value ω', as far as ω' is close to ω, the same random value R can still be extracted. Considering the inputs (ω) to fuzzy sketches and extractors as biometric data, one can utilize these schemes for template protection and biometric-enabled cryptographic applications. Formal definitions for fuzzy sketches and fuzzy extractors will be provided in the following subsections.

The basic security models for fuzzy sketches and extractors [32, 33] and constructions providing security in such models, as reviewed in D5.1, have the following limitations:

Robustness problem. The basic security definitions for fuzzy sketches and fuzzy extractors [32, 33] implicitly assume that attackers cannot modify the (public) helper data, P, associated with an initially enrolled template ω_0. When adversaries can be active and tamper with the helper data during storage or communication, for

instance, in applications such as biometric based remote authentication over insecure channels (where the helper data is sent by the server to the client over an adversarial controlled channel) then the basic known constructions for fuzzy sketches and extractors will fail to provide security in general. In such application environments with active adversaries, one should utilize stronger primitives called "robust" fuzzy sketches and robust fuzzy extractors, introduced by Boyen et al. [15] and further investigated by Dodis et al. [30, 31].

Reusability problem. Another issue with the basic definitions and constructions of fuzzy sketches and extractors is that they implicitly assume that a (secret) biometric template ω_0 is used only once to generate a sketch P (the helper data) and extract a random (cryptographic) key R. Although, repeated use of the recovery algorithm is allowed using the same P with different rescanned biometric samples ω_i (for $i \geq 1$), the secret template is assumed not to be used more than once to extract sketches and keys; otherwise, the basic constructions will not remain secure *in general* and may drastically lose their security for some specific instantiations of their underlying primitives. To guarantee secure reuse of the same biometric secret for different applications, Boyen [14] strengthened the security models for fuzzy sketches and extractors, and introduced "reusable" fuzzy sketches and reusable fuzzy extractors.

Efficient constructions for *robust* fuzzy (sketches and) extractors and *reusable* fuzzy (sketches and) extractors are provided, respectively, in [15, 30] and [14].

In the following, we first review some preliminary mathematical notations and definition. Then we will review the formal definitions for fuzzy sketches and fuzzy extractors which will be needed when providing the formal definitions for robust and reusable variants of these schemes in the following subsections.

Preliminary Notations and Definitions. We let $\{0,1\}^\ell$ denote the set of strings of length ℓ, $\{0,1\}^*$ denote the set of strings of arbitrary but finite length, and U_ℓ denote the uniform distribution over $\{0,1\}^\ell$.

We consider metric spaces with integer-valued distance functions. A metric space $(\mathcal{M}, \mathtt{dis})$ is a finite set $\mathcal{M}$ with a distance function $\mathtt{dis} : \mathcal{M} \times \mathcal{M} \to \mathbb{Z}^+ \cup \{0\}$ satisfying three properties: (1) $\mathtt{dis}(x,y) = 0 \Leftrightarrow x = y$, (2) $\mathtt{dis}(x,y) = d(y,x)$, (3) $\mathtt{dis}(x,z) \leq d(x,y) + d(y,z)$.

A widely used metric space is the Hamming metric where $\mathcal{M} = \mathcal{F}^n$ for some alphabet $\mathcal{F}$ and $\mathtt{dis}(x,y)$ is the number of positions in which strings x and y differ (called the Hamming distance). For instance, representation techniques such as IrisCode [29] convert images of irises into strings in the Hamming space. Other metric spaces, such as subsets of a large universe equipped with the "set difference metric" and finite length strings with the "edit metric" can also be considered depending on different representations of noisy data [32]. For example, the set difference metric can be used when inputs can be represented by a list of their features, e.g. "minutiae" in a fingerprint.

We assume all logarithms to be base 2. The min-entropy of a random variable X is defined as $\mathbf{H}_\infty(X) = -\log(max_x \Pr[X = x])$. This implies that the chance of success of

an attacker in predicting the exact value of X is at most $2^{-\mathbf{H}_\infty(X)}$.

Considering two, possibly correlated, random variables X and Y, the average min-entropy of X given Y is defined [32] as:

$$\widetilde{\mathbf{H}}_\infty(X|Y) \stackrel{def}{=} -\log\left(\mathbb{E}_{y\leftarrow Y}\left[max_x \Pr\left[X=x|Y=y\right]\right]\right),$$

where $\mathbb{E}$ denotes the expected value. It is worth noticing that the average is taken over values of Y (which is assumed to be known but not controlled by the adversary) while the maximum (worst case) is taken over values of X because prediction of X is adversarial given the value of Y. This implies that, on average, the chance of success of an adversary in predicting the exact value of X given the value of Y is at most $2^{-\widetilde{\mathbf{H}}_\infty(X|Y)}$.

Let X and Y be two probability distributions over $\mathcal{S}$. The statistical distance between them is defined as $\mathbf{SD}(X,Y) = \frac{1}{2}\Sigma_{s\in\mathcal{S}}|\Pr\left[X=s\right] - \Pr\left[Y=s\right]|$.

Definition 1 (Fuzzy Sketch)**.** *An $(m,\widetilde{m},t)$-**fuzzy sketch** over a metric space $(\mathcal{M},d)$ comprises a pair of efficient randomized procedures $(\mathtt{Fsk},\mathtt{Rec})$. The fuzzy sketching algorithm **Fsk**, on input $\omega\in\mathcal{M}$, outputs a sketch or helper data $P\in\{0,1\}^*$. The recovery (correction) procedure **Rec** takes an element $\omega'\in\mathcal{M}$ and a helper string $P\in\{0,1\}^*$ as inputs and outputs a string ω'' such that the following two properties are satisfied:*

- *(**Security**) For any random variable W over $\mathcal{M}$ with min-entropy $\mathbf{H}_\infty(W)\geq m$, the average min-entropy of W given the helper value P is $\widetilde{\mathbf{H}}_\infty(W|P)\geq\widetilde{m}$. (I.e., on average, the attacker's chance of success in predicting the secret value of a template giving its associate helper data is at most $2^{-\widetilde{m}}$. The value $m-\widetilde{m}$ is called the entropy loss of the sketch; the less this loss is, the more secure the sketch will be.)*
- *(**Error-tolerance**) If $\mathtt{dis}(\omega,\omega')\leq t$ and $\mathtt{Fsk}(\omega)\rightarrow P$, then $\mathtt{Rec}(\omega',P)=\omega''=\omega$. (I.e., the exact value of the original template ω is recoverable as far as the presented sample ω' is close to the original one.)*

Definition 2 (Fuzzy Extractor)**.** *An (m,ℓ,t,ϵ)-**fuzzy extractor** over a metric space $(\mathcal{M},d)$ comprises a pair of efficient randomized procedures $(\mathtt{Gen},\mathtt{Rep})$. The generation algorithm **Gen**, on input $\omega\in\mathcal{M}$, outputs an extracted string $R\in\{0,1\}^\ell$ and a helper string $P\in\{0,1\}^*$. The reproduction procedure **Rep** takes an element $\omega'\in\mathcal{M}$ and a helper string $P\in\{0,1\}^*$ as inputs and outputs a string R', such that the following two properties are satisfied:*

- *(**Security**) For any distribution W over $\mathcal{M}$ with min-entropy m, the string R is statistically indistinguishable from a uniformly random string even conditioned on the value of P. That is, if $\mathbf{H}_\infty(W)\geq m$ and $\mathtt{Gen}(W)\rightarrow(R,P)$, then we have*

$$\mathbf{SD}((R,P),(U_\ell,P))\leq\epsilon.$$

- *(**Error-tolerance**) If $\mathtt{dis}(\omega,\omega')\leq t$ and $\mathtt{Gen}(\omega)\rightarrow(R,P)$, then $\mathtt{Rep}(\omega',P)=R'=R$. (I.e., the exact value of the (secret) random string R can be reproduced using the helper data P and any new sample ω' which is close to the originally sampled template ω from which R and P were generated.)*

3.1 Robust Fuzzy Extractors

Note that the aforementioned definitions for fuzzy sketches and extractors do not say anything about the case that an adversary may modify the helper string P to another string P'; that is, there is no guarantee about the output of $\mathtt{Rec}(\omega', P')$ and $\mathtt{Rep}(\omega', P')$ when $P' \neq P$.

The notion of "robust" fuzzy extractors, put forth by Boyen et al. [15] and revisited by Dodis et al. [30], provides strong guarantee against such attackers. In a robust fuzzy extractor the reproduction procedure $\mathtt{Rep}$ can output either a string R or a special error code "$\perp$" to signify *failure*. It is required that any modified value $P' \neq P$ forged by an adversary given a genuinely produced value P (using the sample ω) results in $\mathtt{Rep}(\omega', P') = \perp$, for any ω' such that $\mathtt{dis}(\omega, \omega') \leq t$.

In other words, any helper data tampered by an attacker will be detected and rejected by the reproduction algorithm as long as the amount of (possibly adversarial) error (noise) in different samples of the same biometric data remains within the legitimate level (i.e. bounded by t).

Formally, we have the following definitions for *robust* fuzzy sketches and *robust* fuzzy extractors which strengthen the basic definitions of fuzzy sketches and fuzzy extractors by adding the robustness property.

Definition 3 (Robust Fuzzy Sketch)**.** *A fuzzy sketch* $(\mathtt{Fsk}, \mathtt{Rec})$ *over a metric space* $(\mathcal{M}, d)$ *is said to be an* $(m, \widetilde{m}, t, \delta)$-***robust fuzzy sketch*** *if it is an* $(m, \widetilde{m}, t)$-***fuzzy sketch*** *(as in Definition 1) which additionally satisfies the following properties:*

Well-formed*. The recovery algorithm* $\mathtt{Rec}$ *either outputs a string or an error (failure) code "*$\perp$*" such that for all* $\omega' \in \mathcal{M}$ *and arbitrary* P'*, if* $\mathtt{Rec}(\omega', P') \neq \perp$ *then* $\mathtt{dis}(\omega', \mathtt{Rec}(\omega', P')) \leq t$*. (I.e., the recovery algorithm either outputs an error code or outputs a string which is at distance* t *from the input* ω'*.)*

Robust*. Let* $\omega, \omega' \in \mathcal{M}$ *be arbitrary values such that* $\mathtt{dis}(\omega, \omega') \leq t$ *and* $\mathbf{H}_\infty(\omega) \geq m$*. For any adversary* $\mathcal{A}$ *the success (forgery) probability of* $\mathcal{A}$ *in the following game (between a challenger and the adversary) must be at most* δ*: Challenger computes* $\mathtt{Fsk}(\omega) \to P$*, and* $\mathcal{A}(P) \to P'$*; the adversary succeeds in this game if* $P' \neq P$ *and* $Rec(\omega', P') \neq \perp$*.*

Definition 4 (Robust Fuzzy Extractor)**.** *Given algorithms* $(\mathtt{Gen}, \mathtt{Rep})$ *and values* $\omega, \omega' \in \mathcal{M}$*, such that* $\mathtt{dis}(\omega, \omega') \leq t$ *and* $\mathbf{H}_\infty(\omega) \geq m$*, we say that* $(\mathtt{Gen}, \mathtt{Rep})$ *is a (strong)* $(m, \ell, t, \epsilon, \delta)$-***robust fuzzy extractor*** *if it is an* (m, ℓ, t, ϵ)-***fuzzy extractor*** *and, for any adversary* $\mathcal{A}$*, the probability of success (forgery) in the following game is at most* δ*: Challenger computes* $\mathtt{Gen}(\omega) \to (R, P)$*, and* $\mathcal{A}(R, P) \to P'$*; the adversary succeeds in this game if* $P' \neq P$ *and* $Rep(\omega', P') \neq \perp$*.*

A weaker variant, namely "weakly" robust fuzzy extractor, can also be defined similarly with the only difference that in the game the adversary is only given P as the input (instead of both P and R as in the above definition).

We also note that more general variants can be deduced from Definition 4. For example, one can consider an attack game where the adversary is allowed to output more than one modified helper data, say n strings $(P_1, \cdots, P_n)$ and to get outputs of $\mathtt{Rep}(\omega_i', P_i)$, for $1 \leq i \leq n$ (such that $\mathtt{dis}(\omega_i', \omega) \leq t$). Then the adversary succeeds (in breaking the robustness) if there is an i such that $\mathtt{Rep}(\omega_i', P_i) \neq \perp$. It is straightforward to show, using the union bound, that the success probability of the adversary in this game increases at most by a multiplicative factor of n. The same argument also applies for extending Definition 3 to the case where the adversary may try n different helper data.

3.2 Reusable Fuzzy Extractors

The standard security notion of (robust) fuzzy extractors does not consider a setting where the same secret template is used multiple times. In other words, it is assumed that the generation algorithm `Gen` is executed only once on the secret template ω to generate the secret R and the fuzzy sketch P. Recall that the template ω from which the public sketch is created may slightly differ for each execution of `Gen`, and hence an adversary might obtain a large amount of information about the secret template ω. More formally, the conditional min entropy of the secret template ω may drop to 0 after given many values for P that were generated from perturbed templates $\hat{\omega}$.

To overcome these limitations Boyen [14] introduced the notion of *reusable fuzzy extraction*. Informally, a reusable fuzzy extractor guarantees that even after polynomially many executions of the generation function `Gen` (with possibly correlated secret templates ω) no adversary obtains sufficient information about the secret ω. This notion is called *reusable fuzzy extraction* with security against *outsider attacks* and is formalized with respect to a family $\mathcal{F}$ of efficiently computable perturbation functions. More precisely, the outsider attacking game played between an adversary $\mathcal{A}$ and the challenger is described by the following three phases:

Initialization: The challenger selects a random sample ω^* from the distribution W defined over the template space $\mathcal{M}$. Let in the following $(P^*, R^*) \leftarrow \mathtt{Gen}(\omega^*)$ be the target values in the game.

Query phase: The adversary $\mathcal{A}$ can request polynomially many queries of the generation procedure `Gen` by adaptively sending perturbation functions $f \in \mathcal{F}$ to the challenger. The challenger computes the perturbed secret $\hat{\omega}^* = f(\omega^*)$ and computes $\mathtt{Gen}(\hat{\omega}^*) \rightarrow (\hat{R}, \hat{P})$. The value $\hat{R}$ is discarded by the challenger, while the public value (including the sketch) $\hat{P}$ is sent to the adversary $\mathcal{A}$.

Challenge: At the end of the query phase the adversary produces a guess R and wins the game if $R = R^*$.

A fuzzy extractor is *reusable* against outsider attacks if the advantage of winning the above game is negligible in the security parameter. This guarantees that with overwhelming probability no adversary can guess the target secret key R^*. Notice that the above definition

only guarantees the unpredictability of R^*. Boyen also considers a stronger notion of outsider security, where the adversary shall not learn *any* information about the secret template. This security property is captured by an indistinguishability-based security notion [14].

Boyen considers the family of perturbation functions $\mathcal{F}$ defined as follows. For any function $f \in \mathcal{F}$ it holds that, for all $\omega \in \mathcal{M}$, $\mathtt{dis}(\omega, f(\omega)) \leq d$, where d corresponds to the error-tolerance of the underlying fuzzy sketch $(\mathtt{Fsk}, \mathtt{Rec})$ defined in Section 3.1. While this definition of perturbation functions may not be realistic if $\mathcal{A}$ is allowed to tamper with the secret ω, it captures the realistic scenario where the user applies his biometric in many different applications running each time the generation function `Gen` with different but closely related secrets.

Notice that in the previous definition of an outsider adversary the attacker never gets access to the internal secrets R that are generated by the generation `Gen` and reproduction `Rep` procedure. A natural strengthening of outsider security is to allow the adversary additionally to use the public outputs P_i he obtained by running `Gen` on perturbed secrets $f_i(\omega^*)$ and to let him learn the resulting reproduced secret values. This is captured by adding to the above game an additional *private query phase*, where the adversary chooses a perturbation function $f_i \in \mathcal{F}$ and a public string P_i (either freshly chosen or one of the public values obtained during the public query phase) and obtains back the output of `Rep` on input $(f(\omega^*), P_i)$. In the challenge phase, the adversary is supposed to guess $\mathtt{Rep}(\omega^*, P^*)$ for some P^* that has not been queried in one of the private queries.

Boyen proposes two constructions of reusable fuzzy extractors. His first construction resembles the permutation based construction of Dodis et al. [32] and achieves security in the information-theoretic setting against outsider attacks if the underlying secure sketch uses the Hamming metric as the distance measure. The second construction of [14] uses the random oracle model and achieves security against insider attacks. In the random oracle model [9] it is assumed that hash functions behave as perfectly random functions. While random oracles do not exist in practice, proofs in the random oracle model are a first important step to show the soundness of cryptographic schemes. More precisely, any adversary that breaks a scheme with provable security in the random oracle model has to somehow exploit the internal structure of the hash function. It remains an important open problem for future research to develop reusable fuzzy extractors against insider attacks in the standard model, i.e., without relying on the random oracle methodology. Notice that such constructions will necessarily rely on computational assumptions since it is impossible to extract polynomially many secrets from a single source of bounded min-entropy.

4 Attacks and Vulnerabilities

In this section we first highlight some attacks and threats to existing distributed biometric authentication protocols that we presented in Section 2.9. Then, we give a review of security and privacy gaps of advanced privacy-preservation systems.

4.1 Attacks

In [93], the authors present a framework for analysing security and privacy of biometric templates in biometric authentication systems along with several attacks on the protocols described in Section 2.9. Some related attacks are also presented in [4]. Here we list some of them.

Hill Climbing Attack: In this attack the adversary $\mathcal{A}$ uses a fake biometric template and tries to pass the biometric authentication process. More precisely, $\mathcal{A}$ iteratively changes the fake biometric template and keeps only the changes that improve the acceptance score until the fake biometric template will be accepted by the biometric authentication process [68, 87]. It is implied that the biometric authentication system provides an output score that can be exploited by the adversary $\mathcal{A}$ in order to get information for the genuine biometric template.

Below we describe some attacks that can be mounted against a biometric authentication system and can be considered as subcategories of the *hill climbing* attack.

- **Center Search Attack [93] :** The sensor $\mathcal{S}$ is compromised. In the following description $\mathcal{S}$ and $\mathcal{A}$ are used interchangeably. The adversary $\mathcal{A}$ tries to disclose the biometric template b_i that is stored in the database $\mathcal{DB}$. We remind that the sensor $\mathcal{S}$ knows the fresh biometric sample b'_i.

  The attack is divided into the following steps:

  - Step 1: $\mathcal{S}$ flips the first bit of b'_i and sends the resulting sample to $\mathcal{AS}$. $\mathcal{AS}$ performs the whole authentication procedure. If the authentication is successful then the adversary $\mathcal{A}$ proceeds to *step 2*.
  - Step 2: $\mathcal{S}$ flips the second bit of b'_i, leaves the first bit flipped and sends the resulting sample to $\mathcal{AS}$. $\mathcal{AS}$ performs the whole authentication procedure. If the authentication is successful then the adversary $\mathcal{A}$ repeats the same procedure for the third bit and all the following bits until the biometric sample b'_i is rejected. Then, $\mathcal{A}$ proceeds to *step 3*.
  - Step 3: $\mathcal{S}$ restores the first bit of the biometric sample b'_i and sends the resulting sample again to $\mathcal{AS}$. If it is accepted $\mathcal{A}$ can deduce that the first bit of b_i is the same with the first bit of b'_i. Otherwise, the first bits are different.

  By repeating the *step 3*, $\mathcal{A}$ can correct all the bits of b'_i that are different from b_i.

- **Attack 1:** In this attack we consider that the authentication server $\mathcal{AS}$ is malicious and colludes either with the user $\mathcal{U}_i$ or the sensor $\mathcal{S}$. In case $\mathcal{AS}$ colludes with the user $\mathcal{U}_i$ he simulates the sensor $\mathcal{S}$. In both cases $\mathcal{AS}$ performs the *center search* attack in order to recover b_i.

- **Attack 2:** In this attack the authentication server $\mathcal{AS}$ is compromised. $\mathcal{AS}$ receives from the database $\mathcal{DB}$ the encrypted value of a stored reference biometric template

(i.e. $\mathsf{Enc}(b_i)$)[3]. $\mathcal{AS}$ stores the encrypted reference biometric template and generates a fake fresh biometric template b_i'' close to b_i [4]. Then, $\mathcal{AS}$ encrypts b_i'' (i.e. $\mathsf{Enc}(b_i'')$) and performs a *center search* attack to get b_i.

- **Attack 3:** In this attack the authentication server $\mathcal{AS}$ is compromised. $\mathcal{AS}$ waits until a valid fresh biometric b_i' is presented in the sensor $\mathcal{S}$ and accepted in the authentication process. $\mathcal{AS}$ stores the encrypted value of b_i' (i.e. $\mathsf{Enc}(b_i')$) that he receives at some point from the sensor $\mathcal{S}$ and then generates a fake biometric $b_{\mathcal{AS}}$ (using the same assumption as in attack 2). He encrypts this fake biometric template (i.e. $\mathsf{Enc}(b_{\mathcal{AS}})$) and acts as if the $\mathsf{Enc}(b_{\mathcal{AS}})$ comes from the $\mathcal{DB}$. $\mathcal{AS}$ uses $\mathsf{Enc}(b_{\mathcal{AS}})$ in order to complete the authentication process and waits for the matcher's $\mathcal{M}$ decision. Eventually (after $\frac{1}{FMR}$ trials on average), $\mathcal{AS}$ finds a $b_{\mathcal{AS}}$ that is acceptable by the matcher $\mathcal{M}$ and thus, close enough to the original stored reference template b_i. Then, using the *center search* attack $\mathcal{AS}$ will be able to extract b_i'.

- **Attack 4 [4]:** In this attack the authentication server $\mathcal{AS}$ is compromised. We recall that in the Bringer *et al.* protocol the authentication server $\mathcal{AS}$ sends $\mathsf{Enc}(t_j)$ to the database $\mathcal{DB}$ and receives $m_{\mathcal{DB}} = \prod_{j=1}^{N} \mathsf{Enc}(t_j)^{b_{j,k}}$ where $1 \leq j \leq N$ and $1 \leq k \leq M$. $\mathcal{AS}$ computes:

$$\frac{m_{\mathcal{DB}}}{\prod_{j\in J} \mathsf{Enc}(t_j)} = \frac{\prod_{j=1}^{N} \mathsf{Enc}(t_j)^{b_{j,k}}}{\prod_{j\in J} \mathsf{Enc}(t_j)} = \frac{\prod_{j\in J} \mathsf{Enc}(t_j)^{b_{j,k}} \prod_{j\notin J} \mathsf{Enc}(t_j)^{b_{j,k}}}{\prod_{j\in J} \mathsf{Enc}(t_j)}, \tag{2}$$

for $J \subseteq \{1, \ldots, N\}$. If $J = \{j; b_{j,k} = 1\}$ then:

$$\frac{m_{\mathcal{DB}}}{\prod_{j\in J} \mathsf{Enc}(t_j)} = 1.$$

Thus, the authentication server may perform algorithm 1 for attack 4 in order to deduce all $\{j; b_{j,i} = 1\}$ where $1 \leq j \leq N$ and $1 \leq i \leq M$ and thus, to recover multiple bits of the biometric reference templates for multiple users, stored in the database $\mathcal{DB}$. This attack requires 2^N iterations, so it works if N is not too large.

Algorithm 1 Attack 4

For each J where $J \subset \{1 \ldots N\} \backslash i$:
  Compute $\prod_{j\in J} \mathsf{Enc}(t_j)$
  Until $\prod_{j\in J} \mathsf{Enc}(t_j) = m_{\mathcal{DB}}$
Return j

[3]This might be the response to a fake query of $\mathcal{AS}$ to $\mathcal{DB}$.

[4]We assume that $\mathcal{AS}$ has access to a template generator and by exploiting the false matching rate (FMR) of the biometric authentication system he finds one fake template that passes the authentication control after $\frac{1}{FMR}$ trials on average.

- **Attack 5 [93]:** (Against the Barbosa *et al.* protocol).

  The $\mathcal{DB}$ stores the support vectors $\mathsf{SV}_{i,j}$ and the weight coefficients $\alpha_{i,j}$, which are necessary to represent the hyperplanes and thus, perform the classification of new samples.

  We recall that the classification decision is performed by calculating:

$$\begin{aligned} cl^{(j)}_{\mathsf{SVM}} &= v_1 \sum_{i=1}^{S} \alpha_{i,j}(\mathsf{SV}_{i,j})_1 + \ldots + v_k \sum_{i=1}^{S} \alpha_{i,j}(\mathsf{SV}_{i,j})_k \\ &= v_1\beta_{j,1} + \ldots + v_k\beta_{j,k}. \end{aligned} \tag{3}$$

  In this attack we consider that the authentication server $\mathcal{AS}$ is compromised. The adversary performs the following steps:

  1. $\mathcal{AS}$ sends to $\mathcal{DB}$ the vector $v = \langle 1, 0, \ldots, 0\rangle$ and thus, will receive from $\mathcal{DB}$ the encryption of $\beta_{j,1} = \sum_{i=1}^{S} \alpha_{i,j}(\mathsf{SV}_{i,j})_1$
     Apply this strategy (of step (i)) to get the encryption of each β.
  2. $\mathcal{AS}$ instead of sending to $\mathcal{M}$ all c_j for $j \in [1, N]$ he sends:
     - $c_1 = \mathsf{Enc}^{\mathsf{Paillier}}(\beta_{1,1})$
     - $c_2 = \mathsf{Enc}^{\mathsf{Paillier}}(x)$ where $x \in \mathbb{Z}_n$
     - $c_{j'} = \mathsf{Enc}^{\mathsf{Paillier}}(0)$ for $j' \in [2, N]$
  3. $\mathcal{M}$ will return the class j where $j \in [1, N]$ with the greatest value using equation (3). Thus, he will return either 1 or 2 depending on if $\beta_{1,1} > x$ or $\beta_{1,1} \leq x$ correspondingly. Using steps (ii) and (iii) and by employing a dichotomic search the adversary is able to do a $\mathsf{Paillier}$ decryption of anything.

- **Attack 6 [93]:** (Against the Stoianov protocol)

  In this attack, we consider that the matcher $\mathcal{M}$ is compromised. If $\mathcal{M}$ observes the codeword c that is revealed after each such successful authentication, $\mathcal{M}$ is able to trace a legitimate user by checking all his successful authentications.

  In case $\mathcal{M}$ colludes with other entities of the biometric authentication system we discriminate the following cases:

  - $\mathcal{M}$ colludes with $\mathcal{DB}$: In this case $\mathcal{M}$ and $\mathcal{DB}$ are able to trace $c \oplus b$ (linkability attack).
  - $\mathcal{M}$ colludes with $\mathcal{S}$: They control the fresh biometric b' by setting $b' = 0$ they recover $c \oplus b$ (linkability attack).
  - $\mathcal{M}$ colludes with $\mathcal{AS}$. They can always recover b' and thus $c \oplus b$ (linkability attack).

- **Attack 7 [93]:** (Against the Stoianov protocol) Let us assume that $\mathcal{AS}$ is compromised and knows $S' \oplus b'$ that is accepted by the matcher $\mathcal{M}$. Then, $\mathcal{AS}$ can start from $S' \oplus b'$ and add progressively some errors until he gets a rejection from the matcher $\mathcal{M}$. As soon as he gets a rejection he backtracks (puts back one of the changed bits) to get back to a positive result. By the end of this process $\mathcal{AS}$ gets an encrypted template $S' \oplus b''$. Using this encrypted template $\mathcal{AS}$ starts from *phase 2* of the protocol and gets from the database $\mathcal{DB}$ the vector $z' = S' \oplus b'' \oplus \check{S} \oplus c \oplus b$. $\mathcal{AS}$ replaces the first ℓ bits of z'. He tries multiple combinations of the first ℓ bits and sends the resulting vector to $\mathcal{M}$. If for multiple combinations of the ℓ bits the resulting z' vector is accepted then $\mathcal{AS}$ increases the length of ℓ and tries again until for only one combination of the ℓ bits he gets acceptance from $\mathcal{M}$. If this happens then $\mathcal{AS}$ has recovered the first block of $S' \oplus S \oplus c$. By increasing the length of ℓ he is able to recover more bits of $S' \oplus \check{S} \oplus c$. Following this strategy he will be able to recover all the bits of $b' \oplus b$.

4.2 Reflections on Privacy-Preservation Systems

As we have seen in earlier sections, in recent years, homomorphic encryption systems are also used in biometrics. However, the proposed protocols require many rounds of online communication as well as computationally expensive operations on homomorphically encrypted data. Due to these restrictions, the proposed protocols cannot be deployed in practical large-scale applications.

For instance, in the literature, Erkin *et al.* [35] proposed a privacy-preserving face recognition system. More precisely, they proposed a system which employs an eigenface recognition algorithm [98] and designed a protocol that performs operations on encrypted images by means of homomorphic encryption schemes, more concretely, Paillier [78]. They demonstrate that privacy-preserving face recognition is possible in principle and give possible choices of parameter sizes to achieve a good classification rate. However, the proposed protocol requires $O(logN)$ rounds of online communication as well as computationally expensive operations on homomorphically encrypted data to recognize a face in the database of N faces. Due to these restrictions, the proposed protocol cannot be deployed in practical large-scale applications. Then, Sadeghi *et al.* improved the efficiency of this system [89]. However, the system's recognition performance is still limited with the performance of the eigenface method.

Barni *et al.* proposed a privacy-preserving protocol for fingerprint identification using FingerCodes [5, 7]. FingerCodes use texture information from a fingerprint to compare two biometrics. The algorithm is not as discriminative as minutiae based fingerprint matching techniques but it was chosen by the authors as particularly suited for efficient realization in the privacy preserving framework. The proposed protocol ensures security against semi-honest adversaries whereas fails to provide security against malicious adversaries. Homomorphic encryption protects all channels of communication between the server and client by continually re-randomizing the encryption on the data. However, the proposed scheme can be compromised when an attacker directly targets the database because the

database is stored in plaintext. On the other hand, they achieve better performance than [35, 89] in terms of bandwidth and time efficiency.

There are also some works on secure Hamming distance calculation by using cryptographic primitives [62, 76, 82]. However, these papers are limited only to secure Hamming distance calculation. They do not take into account biometric authentication as a whole and fail to satisfy security, privacy, template protection at the same time by taking into account computational efficiency which is very critical for real-world applications. Osadchy *et al.* [76] proposed a scheme called SCiFI based on the Paillier homomorphic encryption and secure Hamming distance calculation for face biometrics. Although they claim that SCiFI is computational efficient, it mostly uses pre-computation techniques. Its pre-computation time includes processing time that must be done locally by each user before using the system. They report that SCiFI's online running time takes 0.31 seconds for a face vector of size 900 bit however its offline computation time takes 213 seconds. Since these computations should be done by a user just before each attempt to use the system, the protocol is not very efficient. Besides, SCiFI is only secure for semi-honest adversaries whereas it is not secure for malicious adversaries. Rane *et al.* [82] also proposed an approach based on secure Hamming distance calculation for biometric applications. However, their proposed method fails to ensure biometric database security since biometric templates are stored in plain format in the database. Thus, a malicious verifier can threat a user's security and privacy. Kulkarni *et al.* [62] proposed a biometric authentication system based on *somewhat* homomorphic encryption scheme. In this system, the user first extracts and sends her biometric features to the trusted enrollment server. Thus, this system uses a trusted enrollment server and fails to protect security and privacy of a user against a malicious database manager. In addition, the system is not computationally efficient since the computation time of a successful authentication is 58 seconds for a 2048-bit binary feature vector.

In an ideal case, the advanced privacy preservation system should perform the authentication process on encrypted data and should not allow any leakage of information about the biometric features. It should achieve secure and private authentication and also provide template protection without loss of accuracy of the matching algorithm. The system should be secure against malicious adversaries. However, many existing schemes present serious limitations and the development of an efficient privacy-preserving biometric authentication protocol can be challenging.

5 Conclusions

In this deliverable we presented an overview of advances in privacy-preserving biometric authentication systems. We specifically focused on privacy-preserving biometric authentication systems that rely on secure multi-party computation. In particular, we reviewed how oblivious transfer, garbled circuits, homomorphic encryption as well as fuzzy extractors can be used in connection with biometric authentication systems to provide security and privacy. Moreover, recently proposed privacy-preserving biometric authentication protocols

using homomorphic cryptosystems are also presented. Finally, we highlighted security and privacy gaps in advanced privacy-preserving techniques, and also explored some attacks on the existing distributed biometric authentication protocols.

References

[1] M. v. d. V. A. Stoianov A., T. Kevenaar. Security issues of biometric encryption. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pages 34–39. IEEE, 2009.

[2] A. Adler. Vulnerabilities in biometric encryption systems. In *In International Conference on Audio and Video based Biometric Person Authentication*, pages 1100–1109, 2005.

[3] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM Conference on Computer and Communications Security*, pages 535–548. ACM, 2013.

[4] M. Barbosa, T. Brouard, S. Cauchie, and S. M. de Sousa. Secure biometric authentication with improved accuracy. In Y. Mu, W. Susilo, and J. Seberry, editors, *ACISP*, volume 5107 of *LNCS*, pages 21–36. Springer, 2008.

[5] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-preserving fingercode authentication. In *Proceedings of the 12th ACM workshop on Multimedia and security*, pages 231–240, 2010.

[6] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida 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. ACM, 2010.

[7] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, and F. Scotti. A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*, Sept. 2010.

[8] D. Beaver. Precomputing oblivious transfer. In D. Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.

[9] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[10] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In V. Atluri and C. Díaz, editors, *ESORICS*, volume 6879 of *Lecture Notes in Computer Science*, pages 190–209. Springer, 2011. Extended version available at `http://eprint.iacr.org/2010/627`.

[11] M. Blum and S. Goldwasser. An efficient probabilistic public key encryption scheme which hides all partial information. In *Proceedings of Advances in Cryptology - CRYPTO'84*, pages 289–299. Springer-Verlag, 1985.

[12] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In R. Dingledine and P. Golle, editors, *Financial Cryptography*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[13] T. E. Boult, W. J. Scheirer, and R. Woodworth. Revocable fingerprint biotokens: Accuracy and security analysis. In *CVPR*, 2007.

[14] X. Boyen. Reusable cryptographic fuzzy extractors. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 82–91. ACM, 2004.

[15] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 2005.

[16] J. Bringer, H. Chabanne, G. D. Cohen, B. Kindarji, and G. Zémor. Optimal iris fuzzy sketches. *CoRR*, abs/0705.3740, 2007.

[17] J. Bringer, H. Chabanne, M. Izabachène, D. Pointcheval, Q. Tang, and S. Zimmer. An application of the goldwasser-micali cryptosystem to biometric authentication. In *Proceedings of the 12th Australasian Conference on Information Security and Privacy*, ACISP'07, pages 96–106, Berlin, Heidelberg, 2007. Springer-Verlag.

[18] J. Bringer, H. Chabanne, and A. Patey. SHADE: Secure hamming distance computation from oblivious transfer. In A. A. Adams, M. Brenner, and M. Smith, editors, *Financial Cryptography Workshops*, volume 7862 of *Lecture Notes in Computer Science*, pages 164–176. Springer, 2013.

[19] J. Bringer, M. Favre, H. Chabanne, and A. Patey. Faster secure computation for biometric identification using filtering. In A. K. Jain, A. Ross, S. Prabhakar, and J. Kim, editors, *ICB*, pages 257–264. IEEE, 2012.

[20] F. M. Bui, K. Martin, H. Lu, K. N. Plataniotis, and D. Hatzinakos. Fuzzy key binding strategies based on quantization index modulation (qim) for biometric encryption (be) applications. *IEEE Transactions on Information Forensics and Security*, 5(1):118–132, 2010.

[21] E.-C. Chang, R. Shen, and F. W. Teo. Finding the original point set hidden among chaff. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 182–188, 2006.

[22] K. H. Cheung, A. W.-K. Kong, J. You, and D. Zhang. An analysis on invertibility of cancelable biometrics based on biohashing. In *CISST*, pages 40–45, 2005.

[23] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, Mar. 2002.

[24] I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 416–430. Springer, 2007.

[25] I. Damgård, M. Geisler, and M. Krøigaard. Homomorphic encryption and secure comparison. *IJACT*, 1(1):22–31, 2008.

[26] I. Damgård, M. Geisler, and M. Krøigaard. A correction to 'efficient and secure comparison for on-line auctions'. *IJACT*, 1(4):323–324, 2009.

[27] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.

[28] J. Daugman. How iris recognition works. *IEEE Trans. Circuits Syst. Video Techn.*, 14(1):21–30, 2004.

[29] J. Daugman. How iris recognition works. *IEEE Trans. Circuits Syst. Video Techn.*, 14(1):21–30, 2004.

[30] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. *IEEE Transactions on Information Theory*, 58(9):6207–6222, 2012.

[31] Y. Dodis, J. Katz, L. Reyzin, and A. Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 232–250. Springer, 2006.

[32] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[33] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer, 2004.

[34] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In I. Goldberg and M. J. Atallah, editors, *Privacy Enhancing Technologies*, volume 5672 of *Lecture Notes in Computer Science*, pages 235–253. Springer, 2009.

[35] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, PETS '09, pages 235–253, 2009.

[36] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[37] Y. C. Feng, P. C. Yuen, and A. K. Jain. A hybrid approach for generating secure and discriminating face template. *Trans. Info. For. Sec.*, 5(1):103–117, march 2010.

[38] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[39] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.

[40] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In A. V. Aho, editor, *STOC*, pages 218–229. ACM, 1987.

[41] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[42] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Tasty: tool for automating secure two-party computations. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 451–462. ACM, 2010. Code available at `https://code.google.com/p/tastyproject/`.

[43] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*. USENIX Association, 2011. Code available at `http://www.mightbeevil.com/framework/download.html`.

[44] Y. Huang, L. Malka, D. Evans, and J. Katz. Efficient privacy-preserving biometric identification. In *NDSS*. The Internet Society, 2011.

[45] T. Ignatenko and F. Willems. Secret rate - privacy leakage in biometric systems. In *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 4*, ISIT'09, pages 2251–2255, 2009.

[46] T. Ignatenko and F. M. J. Willems. Information leakage in fuzzy commitment schemes. *IEEE Transactions on Information Forensics and Security*, 5(2):337–348, 2010.

[47] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[48] A. K. Jain, K. Nandakumar, and A. Nagar. Biometric template security. *EURASIP J. Adv. Signal Process*, 2008:113:1–113:17, Jan. 2008.

[49] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Fingercode: A filterbank for fingerprint representation and matching. In *CVPR*, pages 2187–. IEEE Computer Society, 1999.

[50] A. Jarrous and B. Pinkas. Secure hamming distance based computation and its applications. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS*, volume 5536 of *Lecture Notes in Computer Science*, pages 107–124, 2009.

[51] A. Juels and M. Sudan. A fuzzy vault scheme. *IACR Cryptology ePrint Archive*, 2002:93, 2002.

[52] A. Juels and M. Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006.

[53] A. Juels and M. Wattenberg. A fuzzy commitment scheme. pages 28–36. ACM Press, 1999.

[54] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In J. Motiwalla and G. Tsudik, editors, *ACM Conference on Computer and Communications Security*, pages 28–36. ACM, 1999.

[55] C. Karabat and H. Erdogan. A cancelable biometric hashing for secure biometric verification system. In *Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1082–1085, 2009.

[56] C. Karabat, H. Erdogan, and M. K. Mihcak. A face image hashing method based on optimal linear transform under colored gaussian noise assumption. In *Digital Signal Processing (DSP), 2011 17th International Conference on*, pages 1 –6, july 2011.

[57] A. Kholmatov and B. Yanikoglu. Realization of correlation attack against the fuzzy vault scheme, 2008.

[58] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In J. A. Garay, A. Miyaji, and A. Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2009.

[59] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2008.

[60] K. Kommel and C. Vielhauer. Reverse-engineer methods on a biometric hash algorithm for dynamic handwriting. In *Proceedings of the 12th ACM workshop on Multimedia and security*, MMSec '10, pages 67–72, 2010.

[61] A. Kong, K.-H. Cheung, D. Zhang, M. Kamel, and J. You. An analysis of biohashing and its variants. *Pattern Recogn.*, 39:1359–1368, July 2006.

[62] R. Kulkarni and A. Namboodiri. Secure hamming distance based biometric authentication. In *Biometrics (ICB), 2013 International Conference on*, pages 1–6, June 2013.

[63] K. Kummel, C. Vielhauer, T. Scheidat, D. Franke, and J. Dittmann. Handwriting biometric hash attack: a genetic algorithm with user interaction for raw data reconstruction. In *Proceedings of the 11th IFIP TC 6/TC 11 international conference on Communications and Multimedia Security*, pages 178–190, 2010.

[64] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *IACR Cryptology ePrint Archive*, 2008, 2008.

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

[66] Y. Lindell, B. Pinkas, and N. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In R. Ostrovsky, R. Prisco, and I. Visconti, editors, *Security and Cryptography for Networks*, volume 5229 of *Lecture Notes in Computer Science*, pages 2–20. Springer Berlin Heidelberg, 2008.

[67] Y. Lindell, B. Pinkas, and N. P. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 2–20. Springer, 2008.

[68] M. Martinez-Diaz, J. Fierrez-Aguilar, F. Alonso-Fernandez, J. Ortega-Garcia, and J. Siguenza. Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification. In *Proceedings of the IEEE International Conference on Security Technology (ICCST)*, pages 151–159, Lexington, USA, October 2006.

[69] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. Impact of artificial "gummy" fingers on fingerprint systems. *Datenschutz und Datensicherheit*, 26(8), 2002.

[70] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[71] K. Nandakumar, A. K. Jain, and S. Pankanti. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE Transactions on Information Forensics and Security*, 2(4):744–757, 2007.

[72] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In S. R. Kosaraju, editor, *SODA*, pages 448–457. ACM/SIAM, 2001.

[73] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[74] M. Osadchy and B. Moskovich. Illumination invariant representation for privacy preserving face identification. In *IEEE Computer Society and IEEE Biometrics Council Workshop on Biometrics (CVPRW)*, 2010.

[75] 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. IEEE Computer Society, 2010.

[76] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. Scifi - a system for secure face identification. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 239–254, May 2010.

[77] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[78] P. Paillier and D. Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In *Proc. of Asiacrypt'99, Lecture Notes in Computer Science*, pages 165–179. Springer Verlag, 1999.

[79] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2009.

[80] M. G. Qiming Li and E.-C. Chang. Fuzzy extractors for asymmetric biometric representation. In *Proceedings of IEEE Workshop on Biometrics (In association with CVPR)*, 2008.

[81] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[82] S. Rane, W. Sun, and A. Vetro. Secure distortion computation among untrusting parties using homomorphic encryption. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1485–1488, Nov 2009.

[83] C. Rathgeb and A. Uhl. Statistical attack against iris-biometric fuzzy commitment schemes. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 23–30, 2011.

[84] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[85] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 165–179. Academic Press, 1978.

[86] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[87] C. Roberts. Biometric attack vectors and defences. *Computers & Security*, 26:14–25, 2007.

[88] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In D. Lee and S. Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2009. Extended version available at `http://eprint.iacr.org/2009/507`.

[89] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *ICISC*, pages 229–244, 2009.

[90] W. J. Scheirer and T. E. Boult. Cracking fuzzy vaults and biometric encryption. In *in Proceedings of Biometrics Symposium*, pages 1–6, 2007.

[91] T. Schneider. *Engineering Secure Two-Party Computation Protocols - Design, Optimization, and Applications of Efficient Secure Function Evaluation*. Springer, 2012.

[92] T. Schneider and M. Zohner. GMW vs. Yao? efficient secure two-party computation with low depth circuits. In A.-R. Sadeghi, editor, *Financial Cryptography*, volume 7859 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2013.

[93] K. Simoens, J. Bringer, H. Chabanne, and S. Seys. A framework for analyzing template security and privacy in biometric authentication systems. *IEEE Transactions on Information Forensics and Security*, 7(2):833–841, 2012.

[94] K. Simoens, P. Tuyls, and B. Preneel. Privacy weaknesses in biometric sketches. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, IEEE SP'09, pages 188–203, 2009.

[95] A. Stoianov. Cryptographically secure biometrics, 2010.

[96] A. Stoianov. Security of error correcting code for biometric encryption. In *Eighth Annual Conference on Privacy, Security and Trust, PST 2010, August 17-19, 2010, Ottawa, Ontario, Canada*. IEEE, 2010.

[97] M. A. Turk and A. P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Winter 1991.

[98] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–591, 1991.

[99] T. van der Putte and J. Keuning. Biometrical fingerprint recognition: Don't get your fingers burned. In *CARDIS*, pages 289–306, 2000.

[100] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167. IEEE Computer Society, 1986.