Privacy-Aware Processing of Biometric Templates by Means of Secure Two-Party Computation

R. Lazzeretti, P. Failla, and M. Barni

Abstract The use of biometric data for person identification and access control is gaining more and more popularity. Handling biometric data, however, requires particular care, since biometric data is indissolubly tied to the identity of the owner hence raising important security and privacy issues. This chapter focuses on the latter, presenting an innovative approach that, by relying on tools borrowed from Secure Two Party Computation (STPC) theory, permits to process the biometric data in encrypted form, thus eliminating any risk that private biometric information is leaked during an identification process. The basic concepts behind STPC are reviewed together with the basic cryptographic primitives needed to achieve privacy-aware processing of biometric data in a STPC context. The two main approaches proposed so far, namely homomorphic encryption and garbled circuits are discussed and the way such techniques can be used to develop a full biometric matching protocol described. Some general guidelines to be used in the design of a privacy-aware biometric system are given, so to allow the reader to choose the most appropriate tools depending on the application at hand.

Riccardo Lazzeretti
Information Engineering Department, University of Siena, Italy
e-mail: riccardo.lazzeretti@gmail.com

Pierluigi Failla
Information Engineering Department, University of Siena, Italy
e-mail: pierluigi.failla@gmail.com

Mauro Barni
Information Engineering Department, University of Siena, Italy
e-mail: barni@dii.unisi.it

1 Introduction

Our world is becoming increasingly interconnected and by using the Internet we are able to share everything with everyone. Social networks (e.g. Facebook, Linkedin, MySpace, Twitter) whereby people share thoughts, events, photos and videos with friends are the most evident sign of this general trend. It is clear that behind the distribution and storage of such a massive amount of data there are several security and privacy issues. Potentially privacy sensitive data such as our age, health, preferences, locations, politics and religious views are being stored in computers that we do not own and we do not control. Even worse, the data is generally transferred to third parties in plain format (think about uploading photos or videos on Facebook): people believe in the good will of third parties to behave and handle their data in accordance to laws but also according to their own privacy policies that very often people do not know or do not care about. It is clear that these new platforms and networks are extremely vulnerable to private data disclosures. Current ad–hoc security methodologies, combined with a lack of security sometimes astonishing, will lead to more weaknesses as system complexity and the amount of to-be-handled data increase during the coming years. On the other side, laws aiming at protecting private data are continuously emanated, but legal assurance does not provide a complete answer. Once our private data, preferences or other sensitive information have been compromised, it is virtually impossible to "make them private"again. It is vital that privacy and security of sensitive data as well as its subsequent use be guaranteed a priori.

In some cases, privacy and security constraints are very stringent. Think about traveling. Many people in everyday life use airplanes to move around the world and as everyone knows following the September 11th attacks, controls in airports have been intensified. New electronic passports have been introduced for improved border controls containing personal data, including a *face picture* and *fingerprints*. Each time someone takes a flight the above information is made available to the airport staff and to the police for identity check. To exemplify how security and privacy can easily come at odds, let us consider the following scenario. There are two parties, say an Intelligence Agency and a remote controller (for example the security staff of an Airport). The Agency wants to trace the movements of a suspect person. To do so, it exploits some biometric information of the suspect person. In particular it tries to match the biometric sample it owns with the biometric of the people that are going to take a flight. The Agency wants to protect the identity of the suspect person (and hence the biometries stored in its database) while the Airport wants to protect the privacy of the passengers. From the point of view of the client, the question is: *if I am a good guy, why should I reveal my biometric data to other parties?* At the same time flight safety must be assured, and from the point of view of the Agency, any possible measure to reduce the risk should be taken. More generally, we can affirm that the use of biometric data is becoming a common approach to handle people identities (at Disney World Resort in Florida customers use the fingerprint scanning for the clients that own a multiple-days ticket to assure

the not re-usability [21]), thus raising the call for the adoption of stringent privacy protection measures.

Actually, when dealing with biometric data, the trade-off between the security of the system that needs to be protected and the privacy of the users who provides the biometries is not balanced and the privacy constraints are often overlooked for the sake of security. Often government and law enforcement agencies can access personal information to protect public safety and national security: however, abuses of personal information can cause untold harms, wasted resources, and generally lead to the detriment of society. Hence, there is a high demand for technologies that permit to protect the privacy of users while preserving the possibility of performing biometric analysis with the aim of achieving a greater security.

The most obvious and well–known way to secure personal data is to encrypt and store it in a (trusted) database. Such an approach works only when the owner of the data and the party in charge of processing or storing it trust each other, and the goal of the cryptographic module is only to protect the data from a third party. This is not the case in many practical situations where the owner of the to-be-protected data and the party that is in charge of storing or processing it do not trust each other. Possible examples include the storage of biometric information in a central database, the processing of personal (e.g. medical) data for statistical analysis, or the analysis of people behaviours (e.g. log files) for inspection purposes. How is it possible to combine the request for privacy and the need to analyze personal information for a legitimate purpose (possibly in the interest of the data owner itself)?

1.1 Processing Encrypted Signals

An effective and elegant way to answer the above question, it is to process the data while it is encrypted. In the last thirty years[1] the cryptographic community has worked hardly to build a set of tools that allow to compute with encrypted data. Though this may seem an almost impossible task, some solutions have been put forward recently by relying on the use of:

- **Homomorphic Encryption** whereby some algebraic operations are mapped into simple operations to be applied in the encrypted domain,
- **Secure Multi Party Computation - SMPC** where two or more non-trusted parties engage in an interactive protocol to carry out a computation without revealing their own inputs. The special case where only two parties are involved, such as a Client and a Server, is of particular interest for biometric applications and is usually referred to as Secure Two Party Computation (STPC).

Though the possibility of processing encrypted data (mainly by means of homomorphic encryption) has been advanced more than thirty years ago [63], processing encrypted biometric signals poses some new problems due to the peculiarities of

[1] The first mention is in [63] 1978 by Rivest et al.

this kind of data with respect to the type of data commonly encountered in the cryptographic literature, e.g. alphanumeric strings or bit sequences. The most straightforward difference is that biometric signals are often represented by means of real numbers (and processed by means of floating point arithmetic), while all the available cryptosystems work on integer rings. Other important differences include:

- the non-exact nature of biometric signals that can change significantly from a measurement to the other due to the presence of noise, time variability, pose, gesture etc .... This property should be contrasted with the bit-precise nature of the data cryptosystems usually deal with;
- the essential role played by the temporal or spatial structure of signals, in fact the spatio-temporal dependency between samples of the same process is a core peculiarity of signals.
- the large size of many signals such as audio files, still images, and video sequences, that poses very critical constraints on the complexity and storage requirements.

Despite the above difficulties, some recent studies have shown that the application of non-trivial signal processing tools to encrypted signals is practically feasible.

The great majority of cryptographic primitives used to process encrypted signals relies on two basic mechanisms: homomorphic encryption [62] and garbled circuits [69].

Homomorphic cryptosystems have the property that some elementary algebraic operations in the plain domain are mapped into elementary operations in the encrypted domain. For instance, in the Pailler cryptosystem [58], an addition in the plain domain corresponds to a multiplication in the encrypted domain. Other examples of homomorphic cryptosystems include RSA [64] that is multiplicatively homomorphic on product, Damgaard-Jurik's generalization of Pailler's cryptosystem [25] and Bresson et al. cryptosystem [18] (that is additively homomorphic). If a homomorphic cryptosystem is used, it is possible for a party that does not posses the decryption key to perform some simple operations on the encrypted messages.

The current state of the art in homomorphic encryption does not allow the efficient simultaneous preservation of addition and multiplication. As a matter of fact a very recent result by Gentry in [32] shows that algebraically homomorphic cryptosystems are possible, however actually such schemes are of theoretical interest only, given their extremely high complexity. Due to the unavailability of efficient algebraically homomorphic cryptosystems, homomorphic encryption does not allow the application of non-linear operators, which, on the other side, are essential ingredients of most non-trivial operation to be applied to the encrypted signals. To avoid the above limitation, the general approach is to use an interactive protocol whereby a Client and a Server collaborate and exchange data to securely compute a given functionality.

Garbled circuits have been introduced by Yao in 1982 [69] and later refined in [36], where it is shown that any function can be computed in a secure manner by implementing a boolean circuit of secure gates. With Yao's circuit approach one can evaluate circuits in a privacy preserving scenario by using either private-key or

public-key primitives. Approaches based on symmetric primitives are several orders of magnitude faster than the asymmetric approaches. In its basic (two party computation) form, garbled circuits allow two users, namely the Client ($\mathcal{C}$) and the Server ($\mathcal{S}$), to securely evaluate a known function (usually in form of a boolean circuit) using their private inputs. In other words, executing the evaluation protocol does not reveal any knowledge about the inputs beyond what can be deduced merely from the computed output(s). The circuit approach can be relatively efficient in different security models even if it requires to transfer a large amount of data from one party to the other which yields an increase in the communication complexity of the protocol.

1.1.1 Application scenarios

The techniques briefly outlined above have been used in a variety of application scenarios.

Biometry matching is one of the most important topics wherein secure computation can be used. In [27, 28] a privacy-enhanced face recognition system is proposed. The proposed construction is based on homomorphic encryption and allows to hide the biometric data using an encrypted version of Eigenfaces-based matching, and it is able to hide the result of the match to the server that actually performs the matching operation. A different STPC face matching algorithm based on garbled circuits has been proposed in [65]. Similarly, in [57] an ad hoc system for face recognition in a privacy preserving framework is proposed, specifically designed for usage in secure computation.

In [5, 6] the authors consider a scenario where a client equipped with a fingerprint reader is interested to learn if the acquired fingerprint belongs to a database of authorized entities managed by a server. Although privacy-preserving biometric identification usually focuses on selecting the best matching identity in the database, the solution proposed in [5, 6] also allows to select and report all the enrolled identities whose distance to the user's fingercode is under a given threshold.

In remote diagnosis [19] secure computation can be used to preserve the privacy of the patients. In [7] a privacy-preserving system is described whereby the Server classifies an ElectroCardioGram (ECG) signal without learning any information about the ECG signal and the Client is prevented from gaining knowledge about the classification algorithm used by the Server. The system relies on the concept of Linear Branching Programs (LBP) and a related cryptographic protocol for secure evaluation of private LBPs [7, 8] based on homomorphic encryption and garbled circuits. The paper faces with the study of the trade-off between signal representation accuracy and system complexity both from practical and theoretical perspectives. As a result, the inputs to the system are represented with the minimum number of bits ensuring the same classification accuracy of a plain implementation. In [9] the same classification task is addressed by applying a neural network to the encrypted ECG signal. Quality evaluation of the ECG signals in the encrypted domain has been addressed in [10] to avoid that noisy signals are processed returning wrong classification results.

Many other application fields can benefit from privacy preserving techniques. In [30], a novel technique has been proposed to compute the well-known A^* algorithm, on the encrypted weights of a graph. A^* is a *best first* graph search algorithm that uses an heuristic function helping to choose the best candidates during the traversing of common graphs [39]. Graphs are data structures widely used to represent: social networks; computer networks; geographic maps; game moves; possible paths in a given environment and many more, and hence working on encrypted graphs may find several interesting applications. In the setting considered in [39] two parties are interested to compute the shortest path between two nodes in a context where: i) part of the graph topology (only the number of nodes) is publicly known, ii) the client knows the weights of each edge and iii) the Server owns the heuristic used for searching. The Client wants to keep secret the weights and the Server the heuristic used.

In [31], a scenario in which two parties are interested in computing a given functionality in a privacy preserving way has been considered, but this functionality needs a sub–protocol that computes the *Gram – Schmidt Orthogonalization* on encrypted vectors. There are a lot of applications in which this kind of sub–protocol could be used as a basic privacy preserving primitive, including: QR decomposition [38]; linear least squares problems [15]; face recognition [70]; improving performances of neural networks [56]; wavelets computation [22]; principal component analysis [66] and image compression [50].

In [13] the authors analyze the implementation of the Fourier transform in a privacy preserving scenario. In [42] an efficient buyer-seller protocol embedding an encrypted watermark in a content is proposed, protecting the watermark secrets from the buyer and preventing false infringement accusations by the seller.

Other applications include data mining [1, 47]; secure compression [41]; access to encrypted databases [20]; encrypted strings comparison by using Levenshtein distance [61], etc.

1.2 Processing of biometric signals

Most tasks in biometric signal processing are based on pattern recognition, it is hence necessary to develop protocols that permit to apply at least the most basic pattern recognition operators to encrypted signals. The number of tools and tasks usually encompassed under the Pattern Recognition umbrella is virtually endless. An exhaustive discussion of how such tools could be applied in the encrypted domain is then impractical. For this reason we focus our discussion on the pattern recognition task most commonly used in biometric systems, namely *Pattern Matching*. There are two basic forms of Pattern Matching: i) *Verification*, as a *one to one* matching problem, and ii) *Identification*, as a typical *one to many* matching problem.

More specifically, the verification problem can be defined as follows: given two patterns V_1 and V_2 decide whether they represent the same object or not. On the other side, the identification problem answers the following question: given a pattern V

and a set of patterns $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$, is there a pattern in $\mathcal{V}$ that corresponds to V? If yes, which is the index of such a pattern?

From a very general point of view, pattern matching can be seen as a two-step process. The first step is the so called feature extraction step, in which the pattern to be classified is transformed into an (m-long) vector whose components, called features, describe some particular characteristics of the to-be-classified pattern. As a first example, we may consider the classification of image regions. The region to be classified is the *pattern*, while the feature vector may contain the average gray level and the standard deviation of the pixels belonging to the region, the area of the region, its inertia moments, etc. A second example regards the classification of heart beats based on ECG signals. The *pattern* to be classified is the portion of ECG signal corresponding to a single heart beat, while the features may be the coefficients of the AR (autoregressive) model that better describes the signal, or a set of statistics extracted from the ECG.

Feature extraction is a crucial and necessary step since on one side it permits to simplify the pattern description by reducing it to a vector in $\mathbb{R}^m$, and on the other side the extracted features are supposed to describe some meaningful characteristics of the pattern to be classified. After feature extraction the *pattern* to be matched (or classified) is nothing but a vector V belonging to $\mathbb{R}^m$.

The second step is the actual matching step, in which a test pattern V is matched against one or several patterns $\{V_1, V_2, \ldots, V_n\}$.

The feature extraction step is highly application dependent and no general theory exists for it. For this reason, it is not possible to define the set of primitives that need to be developed to extract the features in the encrypted domain. Moreover, the involved operations are usually highly non-linear and their implementation in a STPC framework would be extremely cumbersome. Of course exceptions to this general rule exist, but in the majority of the cases we can assume that features are extracted in the plain domain and then processed securely by means of STPC techniques.

1.3 Goals and Outline of the Chapter

The goal of this chapter is to present some guidelines for developing a biometry matching protocol working in the encrypted domain, regardless of the kind of biometry being analyzed. Specifically, we will discuss several approaches to build the basic modules of any biometric matching protocol (distance computation and minimum selection), and show how such modules can be conveniently used in a great variety of different scenarios.

This rest of the chapter is organized as follows. The biometry matching problem is rigorously defined in Sect. 2. Then we show the cryptographic primitives necessary to the implementation of matching algorithms in the encrypted domain (Sect. 3). In Sect. 4 we describe how such primitives can be used to implement some of the most basic pattern recognition building blocks. In Sect. 5 we give some hints about

how the building blocks can be assembled to carry out a given functionality. Finally in Sect. 6, some conclusions are provided.

2 Biometric template matching

As we said, regardless of the type of biometry and the feature extraction protocol, we can assume that any biometric template V is represented by a vector of m features, each assuming integer value (possibly resulting from a quantization problem) in the set $\{0,\ldots,b-1\}$, i.e. $V \in \mathbb{Z}_b^m$.

2.1 The verification problem

As opposed to feature extraction, processing the feature vectors in a pattern matching application is a rather standard (though not easy) task always following a few number of fixed steps. It is then extremely important that these steps are defined and their privacy requirements identified, since doing so will allow to build a rather general theory. In this Section we start by considering the easiest problem, namely the verification problem: "Is he who he claims to be?".

The general verification problem can be summarized as follows:

- One party, say $\mathcal{C}$ knows a feature vector V_1.
- Another party, say $\mathcal{S}$ knows another feature vector V_2.
- We want to answer the question: is V_1 close enough to V_2?

As it can be seen, the verification problem boils down to only two operations: i) distance calculation and ii) comparison against a threshold. As soon as an efficient protocol is available to perform these two tasks, a secure protocol for pattern verification can be built. In order to do so, it is necessary that the privacy requirements are defined. Though many different scenarios are possible, in most of the cases the requirements of the protocol can be defined as follows (see also the summary in Table 1):

- $\mathcal{C}$ gets yes/no.
- $\mathcal{S}$ gets nothing or yes/no.
- V_1 and V_2 are private inputs of $\mathcal{C}$ and $\mathcal{S}$ respectively and have to be kept secret.
- The distance function and the threshold may be assumed to be public parameters.
- Any intermediate value is not revealed to both parties.

Table 1 V isibility
of the values involved in the verification problem.

	V_1	V_2	Intermediate values	Final result
$\mathcal{C}$	yes	no	no	yes
$\mathcal{S}$	no	yes	no	optional

2.2 The identification problem

While the verification problem involves a *one to one* matching, the identification problem corresponds to a *one to many* match and is used when two or more parties are interested to answer the question "Who is he?".

Specifically, pattern identification can be summarized as follows:

- One party, say $\mathcal{C}$ knows a feature vector V_{test}.
- Another party, say $\mathcal{S}$ knows a set of feature vectors $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$.
- The possible questions to be answered are:

  1. Is V_{test} close to at least one $V_i \in \mathcal{V}$? Boiling down to: is the minimum distance between V_{test} and the elements in $\mathcal{V}$ smaller than a threshold?
  2. Which is the index of the feature vector in $\mathcal{V}$ closest to V_{test}?
  3. How many elements in $\mathcal{V}$ are close enough to V_{test}?

As it can be seen, in most of the cases identification boils down to calculation of several distances, thresholding and/or computation of a minimum. Hence there are two main differences with respect to verification. The first one is quantitative, in that several distances and thresholds must be computed instead of one. The second is qualitative, since a new operator, namely minimum computation is needed.

With regard to the privacy requirements, the situation is slightly more complicated than in the verification case, however it is still possible to define a standard set of requirements (see also Table 2):

- $\mathcal{C}$ gets: i) yes/no or ii) the index of the minimum distance feature vector
- $\mathcal{S}$ gets nothing
- V_{test} (owned by $\mathcal{C}$) and $\mathcal{V}$ (owned by $\mathcal{S}$) are kept secret.
- The distance function and the threshold may be assumed to be public parameters.
- Intermediate values are not revealed to both parties.

A common alternative in which the identification is used by $\mathcal{S}$ to decide whether $\mathcal{C}$ belongs to a set of users allowed to access a given system is the following:

- $\mathcal{C}$ gets nothing or yes/no
- $\mathcal{S}$ gets yes/no
- V_{test} (owned by $\mathcal{C}$) and $\mathcal{V}$ (owned by $\mathcal{S}$) are kept secret.
- The distance function and the threshold may be assumed to be public parameters.
- Intermediate values are not revealed to both parties.

Table 2 V isibility of the values involved in the identification problem.

	V_{test}	V_i	Intermediate values	Final result (yes/no or index)
$\mathcal{C}$	yes	no	no	yes
$\mathcal{S}$	no	yes	no	no

The privacy requirements of the access control problem are shown in Table 3.

Table 3 V isibility of the values involved in the access control problem.

	V_{test}	V_i	Intermediate values	Final result (yes/no)
$\mathcal{C}$	yes	no	no	yes
$\mathcal{S}$	no	yes	no	yes

3 Cryptographic primitives

The problem of computing with encrypted data is a central one in the field of cryptography and goes back to the early days of modern cryptography, about thirty years ago [63]. The problem has a fundamental importance both from a theoretical and a practical perspective. Often and especially in the case of number theoretic cryptosystems, the possibility of computing with encrypted data is a direct consequence of a common property of the cryptosystems: the malleability. More in detail a cryptosystem is malleable if given an encryption of a plaintext m, it is possible to generate another ciphertext which decrypts to $f(m)$, for a known function f, without necessarily knowing or learning m.

Although from a security point of view malleability is a weakness of a cryptosystem because it allows to modify the plaintext using only the ciphertext, in our context it is the key that allows to compute on encrypted data. Starting from the pioneering works of Yao [69] and Goldreich, Micali and Widgerson [36] the problem has been extensively studied in a variety of settings and under different assumptions, up to homomorphic encryption and garbled circuits, the two main tools actually used in SMPC.

As we said in the introduction, a specific case of SMPC particularly interesting for biometric applications is Secure Two Party Computation where only two entities are involved: a server $\mathcal{S}$ (the service provider) and the client $\mathcal{C}$ (a user that needs to access to a functionality provided by the server).

Before introducing the main cryptographic tools available for STPC, we pose to discuss two cornerstones of STPC: security and complexity.

Security. Mistrust among parties is usually modeled by assuming the existence of an adversary that is allowed to corrupt some partial set of the parties, so that the adversary can read (and possibly modify) the internal state of the corrupted players. A possible lack of reliability in the communication is modeled by allowing the adversary to control the communications involving corrupted players. The SMPC paradigm allows many settings and concerns to be modeled and is a strong tool in showing that solutions exist to very general cryptographic problems. The power of the framework is that under the assumption of partial corruption (and various settings and constraints) it is possible to compile any polynomial size function into a protocol that maintains the privacy of the inputs. Input privacy is assured facing an adversary that is assumed to control the entire state (memory) of corrupted parties (passive adversary) and one that in addition may corrupt the memory arbitrarily (active adversary).

In STPC protocols we would like to have the same correctness and reciprocal privacy as in a trusted domain, but this is quite difficult to achieve. For this reason in many cases SRPC developers adopt the *honest but curious* model (also said *semi-honest* model), according to which both parties are considered passive and follow the protocol but try to infer additional information from the transcript of messages seen in the protocol. So the parties may deviate from the protocol only in their internal computation, but the messages are constricted and sent in accordance to the protocol. Far from trivial, this model covers many typical practical settings such as protection against insider attacks. Further, designing and evaluating the performance of protocols in the honest but curious model is a first step toward protocols with stronger security guarantees. Indeed, most protocols for practical privacy-preserving applications focus on the honest but curious model [48]. In most cases sub-protocols are stacked together to obtain more complicated functionalities, in this context it is really important to know that if all sub-protocols are proven secure in the honest but curious model then their sequential composition inherits this security property [35].

Sometimes it is necessary to assume that one party can act maliciously, i.e. he behaves as an active adversary available to cheat to obtain information relative to the other party. For sake of brevity in this chapter we will not cover such scenarios.

Complexity. General multiparty computation protocols allow to securely compute any function but this results in inefficient solutions when compared to plain protocols. For this reason, efficient ad-hoc solutions have to be designed to solve specific cryptographic problems.

We can analyse complexity from three different points of view:

- *Number of Bit Operations*: this is also called computational complexity and indicates the number of basic operations that the protocol needs;
- *Number of Rounds*: the protocols we focus on are client–server protocols, i.e. they require some message exchanges to carry out the computation, a measure of the efficiency of this kind of protocols is the *number* of unidirectional transmission of information they require;
- *Bandwidth*: it is the amount of bits exchanged during protocol execution.

To measure the number of bit operations we use the Big-$\mathcal{O}$ notation [43]. Analyzing some basic operations needed in a STPC scenario and assuming that the largest number involved in the computation is represented by ℓ bits, i.e. the size of a ciphertext is ℓ bits, we have that the cost of an addition between two numbers is $\texttt{add} = \mathcal{O}(\ell)$; that of a multiplication $\texttt{mult} = \mathcal{O}(\ell^2)$; and that of an exponentiation $\texttt{expo} = \mathcal{O}(\ell^3)$. Computing an hash function has a constant complexity that does not depend on ℓ. For the sake of simplicity in the rest of this chapter we will use as measure of computational complexity the operation having the largest complexity among all the operations involved in the protocol.

While the number of rounds is a very simple concept, we spend a few words about the bandwidth. The bandwidth depends on many factors, but probably the most important one is the cryptosystem and so the size of the ciphertext. Some cryptosystems require to transmit only the values used during the evaluation, while others need to transmit other information relative to the functionality, moreover each value is usually represented by long ciphertexts, resulting in big amount of data exchanged between the parties.

3.1 Symmetric vs asymmetric encryption

Cryptographic systems can be divided in symmetric systems, where encryption and decryption are performed by using the same key, and asymmetric systems where the public encryption key and the secret decryption key are distinct.

3.1.1 Symmetric Encryption

Symmetric key cryptography is one of the oldest methods in security systems and provides the confidentiality of a service. The most important properties of these algorithms are ease of operation and high speed [53]. In protocols based on symmetric encryption both the sender and receiver have a common secret key which is used for encryption and decryption of messages. It is assumed that decrypting a message is easy when the key is known and otherwise difficult. Indeed the encryption acts as a secure communication channel between sender and receiver (as shown in Fig. 1) an eavesdropped does not have access to.

There are several kinds of symmetric key systems proposed in the literature and used in practice like triple DES and AES (see [23] and [4]).

3.1.2 Asymmetric Encryption

Symmetric key systems are useful and efficient, but their application requires special infrastructures to be setup. Some examples are setting up the initial keys or managing keys among several users. A solution is using public key encryption in

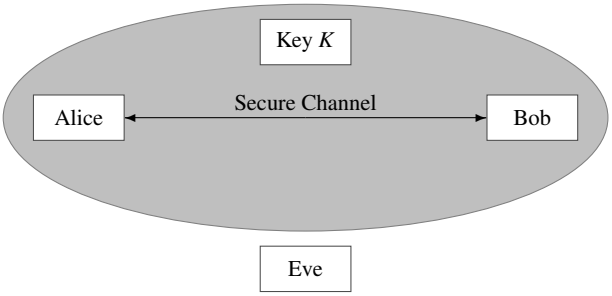


Fig. 1 The typical scenario for symmetric key cryptography. Alice and Bob are legitimate users of the system, whereas Eve is malicious and wants to eavesdrop the channel. The key K is the common secret key between Alice and Bob and creates a secure channel for communication between legitimate users.

which encryption and decryption are performed using two different keys. The first one (public key) is published, whereas the second one (secret key) must be kept secret.

The first efficient public key encryption scheme was the RSA system [64], based on the difficulty of factoring large composite numbers.

An important class of public key cryptosystems are systems based on probabilistic encryption proposed for the first time in [37]. In these systems a given plaintext is encrypted into a different message at each new encryption. This is useful when, e.g., encrypting and transmitting single bits. If the encryption was deterministic, the adversary could encrypt the bits zero and one, and always see which of these encryptions has been transmitted.

The most popular public key cryptosystem with semantic security (IND-CPA) is the Paillier cryptosystem (see Sect. 3.2.1), introduced for the first time in [58] and based on the difficulty of deciding if a number is an n-th power in $\mathbb{Z}_{N^2}$, for a large enough N.

Many public key encryption schemes have homomorphic properties and can be used as cryptographic primitives in SMPC applications.

3.2 Homomorphic encryption

A homomorphic encryption (HE) scheme over an algebraic ring, can allow additive, multiplicative or algebraically homomorphisms. In particular it permits to perform an algebraic operation $\bullet$ between encrypted numbers that returns, after decryption, the same result of the operation $+$ performed on the same values in the plain domain:

$$a+b=\mathcal{D}([\![a]\!]\bullet[\![b]\!])); \tag{1}$$

or permits an operation $\circ$ on ciphertexts such that

$$a * b = \mathcal{D}([\![a]\!] \circ [\![b]\!])); \tag{2}$$

where $[\![\cdot]\!]$ and $\mathcal{D}(\cdot)$ indicate respectively encryption and decryption. In the first case we say that the cryptosystem is additively homomorphic while in the second one it is multiplicatively homomorphic. Finally a cryptosystem is said to be algebraically homomorphic if both operations $\bullet$ and $\circ$ exist such they have a mapping in the algebraic addition and multiplication.

Table 4 shows a list of cryptosystems with their homomorphic properties.

Table 4 Homomorphic Properties and Homomorphic Cryptosystem.

Cryptosystem	Add	Mult	Both
RSA (*1978*, [64])	NO	YES	NO
Goldwasser-Micali (*1982*, [37])	YES	NO	NO
ElGamal (*1985*, [26])	NO	YES	NO
Benaloh (*1994*, [12])	YES	NO	NO
Paillier (*1999*, [58])	YES	NO	NO
Boneh-Goh-Nissim (*2004*, [16])	YES	only 1	YES
Gentry (*2009*, [32])	YES	YES	YES

The most popular homomorphic cryptosystems permit to evaluate the sum among ciphertexts [58], while El-Gamal cryptosystem [26] permits to evaluate the product. For several years the researcher community tried to propose a fully homomorphic cryptosystem with no significant results, but in 2009 in a breakthrough result by Gentry [32, 67], the first fully homomorphic encryption scheme was finally proposed. Gentry's paper shows how to use ideal lattices to construct an encryption scheme that allows to encrypt single bits and that is homomorphic with respect to addition and multiplication. Even though this result is a major theoretical achievement because secure fully homomorphic encryption was suspected to be impossible to achieve [17], the scheme itself and its recent improvements are still too inefficient to be used in practice. Very recently Melchor et al. in [52] and Gentry et al. in [33] have conceived less general forms of homomorphic encryption schemes based on lattices which are more efficient than existing fully homomorphic schemes but still unsuitable for most applications. Such schemes are less general in the sense that they allow only a limited number of multiplications.

3.2.1 Paillier Cryptosystem

The most popular homomorphic cryptosystem, used extensively in several SMPC protocols, is Pailler cryptosystem. To illustrate the way Paillier cryptosystem works, we start by defining its public and private keys generation, the encryption and the decryption.

- *Key generation*: given an RSA modulus $N = pq$ and $\lambda = \text{lcm}(p-1, q-1)$ and selected an integer generator $g \in \mathbb{Z}^*_{N^2}$ such that $N|\text{ord}(g)$ (N divides the order of

g), meaning that:

$$GCD(L(g^{\lambda} \bmod N^2), N) = 1,$$

the public (encryption) key is $PuK = (N, g)$ and the private (decryption) key is $PrK = (\lambda, \mu)$, where $\mu = (L(g^{\lambda} \bmod N^2))^{-1} \bmod N$, and $L(\cdot)$ is an integer function defined by

$$L(u) = \lfloor (u-1)/N \rfloor.$$

- *Encryption*: the encryption of the message $m \in \mathbb{Z}_N$ is $[\![m]\!]$ when

$$[\![m]\!] = g^m r^N \bmod N^2, \tag{3}$$

where $r \in_R \mathbb{Z}_N^*$.

- *Decryption*: given the encryption $[\![m]\!] \in \mathbb{Z}_{N^2}$, the original message m can be obtained as:

$$m = L([\![m]\!]^{\lambda} \bmod N^2)\mu \bmod N.$$

Paillier cryptosystem has several interesting properties. In the following we point out the most important ones.

Given $x, y, k, r \in \mathbb{Z}_N^*$ we have:

Proposition 1. ***Additive Homomorphism.***

- $\mathcal{D}([\![x]\!][\![y]\!] \bmod N^2) = x + y \bmod N$

Proposition 2. ***Scalar Homomorphism.***

- $\mathcal{D}([\![x]\!]^k \bmod N^2) = kx \bmod N$

Proposition 3. ***Self–Blinding.***

- $\mathcal{D}([\![x]\!] r^N \bmod N^2) = x \bmod N$

The security of the Paillier cryptosystem is provided under the Composite Residuosity Problem and Decisional Composite Residuosity Problem. These problems are considered intractable and so suitable as basis for a cryptosystem (a detailed discussion can be found in [58]). It can be also proven that under the assumption that the Decisional Composite Residuosity Problem is untractable, Paillier cryptosystem is a randomized IND-CPA[2] cryptosystem.

In general we indicate with $T = \lceil \log_2 N \rceil$ the Paillier security parameter and we define $2T$ the bit size of a ciphertext. The most updated NIST[3] recommendation for security parameters is to use at least $T = 1024$ (more detail in [3]).

Note that for Paillier cryptosystem the computational complexity is: `enc` $\approx$ `dec` $\approx$ `expo`, hence the computation complexity of a homomorphic protocol can be related to the number of `expo` necessary to its execution.

[2] Indistinguishability under chosen-plaintext attack (IND-CPA) assures that given two plain messages and the encryption of one of them, the adversary, can not identify the message choice with probability significantly better than 1/2

[3] National Institute of Standard and Technology. The mission of the Institute is to: *"promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve quality of life."*

3.2.2 Non linear computation by using blinding

Most protocols require that non linear functions are computed. In a privacy preserving scenario such functions can not be computed by relying on homomorphic properties only and interaction among the parties is required. In those cases $\mathcal{S}$ asks to $\mathcal{C}$ some help to carry out a portion of the computation. This introduces an interaction between the parties during which everything must be kept secret. Formally $\mathcal{S}$ has some data $[\![x]\!]$ encrypted with the public key of $\mathcal{C}$ and needs to compute the functionality $f(\cdot)$ with the help of $\mathcal{C}$. Since $\mathcal{C}$ owns the private key, it is able to obtain x, but $\mathcal{S}$ does not want to reveal it to $\mathcal{C}$, because it can be used to extrapolate some information owned by $\mathcal{S}$. Hence $\mathcal{S}$ chooses a random r and by homomorphic properties computes $[\![x+r]\!]$ and sends it to $\mathcal{C}$. $\mathcal{C}$ decrypts the message obtaining $x+r$ from which it cannot retrieve x. At this point $\mathcal{C}$ computes $[\![f(x+r)]\!]$ and sends it back to $\mathcal{S}$ that obtains the required computation. Obviously, it is necessary that $\tilde{f}(\cdot)$ exists such that

$$\tilde{f}([\![f(x+r)]\!], r) = [\![f(x)]\!].$$

Fig. 2 summarizes the flow of actions in blinding-based SMPC.

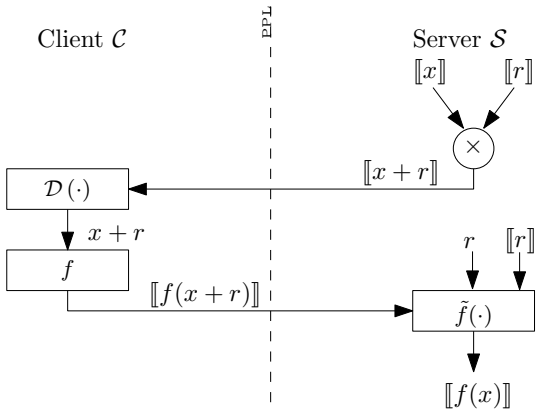


Fig. 2 Blind Computation with Encrypted Data (PPL indicated the Privacy Preserving Line and $\times$ denotes the product).

The security of this kind of schemes stems from the information theoretic security of additive blinding (the mutual information between x and $x+r$ decreases exponentially fast with the number of bits necessary to represent r), so it provides a perfectly secure and practical approach for computing on encrypted data. Additive blinding is used quite often, and several sub–protocols have been developed by using this approach. Here we present the EncMul, EncSquare and BitMin protocols that will be used later in Sect. 4.

- EncMul protocol: to exemplify the blinding procedure outlined above, we now describe the sub–protocol, EncMul, that allows to compute the product of two

Paillier ciphertexts obtaining $[\![xy]\!] = \mathsf{EncMul}([\![x]\!],[\![y]\!])$. Suppose (See Fig. 3) that $\mathcal{S}$ owns $[\![x]\!]$ and $[\![y]\!]$ encrypted with the public key of $\mathcal{C}$. It can obfuscate both chyphertexts by adding two random numbers due to homomorphic additive properties and obtain $[\![x+r_x]\!]$ and $[\![y+r_y]\!]$. Upon reception of $[\![x+r_x]\!]$ and $[\![y+r_y]\!]$, $\mathcal{C}$ decrypts and multiplies them obtaining $w = xy + xr_y + yr_x + r_x r_y$. A this point $\mathcal{C}$ encrypts w and sends it back to $\mathcal{S}$ that computes:

$$\begin{aligned}
[\![w]\!][\![x]\!]^{-r_y}[\![y]\!]^{-r_x}[\![-r_x r_y]\!] &= [\![w]\!][\![-xr_y]\!][\![-yr_x]\!][\![-r_x r_y]\!] = \\
&= [\![w - xr_y - yr_x - r_x r_y]\!] = \\
&= [\![\underbrace{xy + xr_y + yr_x + r_x r_y}_{w} - xr_y - yr_x - r_x r_y]\!] = \\
&= [\![xy]\!] \qquad (4)
\end{aligned}$$

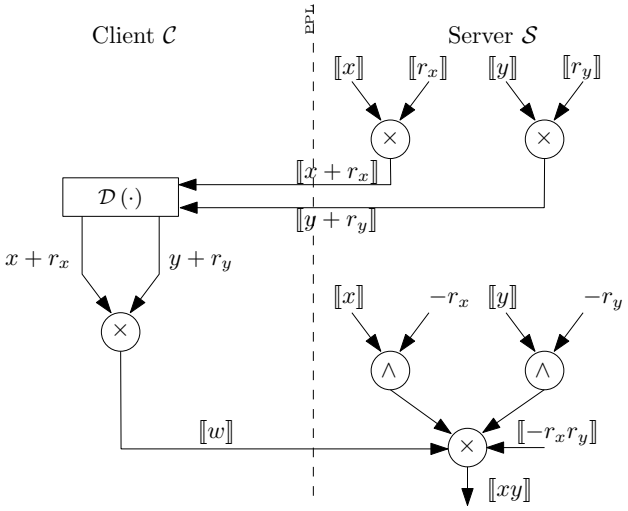


Fig. 3 The protocol EncMul.

Computing EncMul requires 2 rounds (one from $\mathcal{S}$ to send the obfuscated ciphertexts and one from $\mathcal{C}$ to send back the result) and a bandwidth of $3 \times 2T$ bits (3 ciphertexts are exchanged having size $2T$ bits, where T is the Paillier security parameter introduced in 3.2.1) with a computational complexity equal to: 3 `enc` to encrypt the obfuscation values $r_x,\ r_y,\ -r_x r_y$, 2 `expo` needed to compute $[\![x]\!]^{-r_y}$ and $[\![y]\!]^{-r_x}$; 5 `mult` needed to obfuscate $[\![x]\!]$, $[\![y]\!]$ and to compute the additions to $[\![w]\!]$; 2 `dec` to obtain in plain $x+r_x$ and $y+r_y$ and finally 1 `enc` to encrypt the result, for an asymptotic complexity of 8 `expo` operations.

- EncSquare protocol: the EncMul protocol can be optimized to compute the square of a value, resulting in the EncSquare protocol. $\mathcal{S}$ owns $[\![x]\!]$, obfuscates it by adding a random number r_x and obtains $[\![x+r_x]\!]$. At this point $\mathcal{S}$ sends the

cyphertexts to $\mathcal{C}$ that decrypts it, computes the square value $w = (x + r_x)^2$ and sends it back to $\mathcal{S}$, after encryption. Finally $\mathcal{S}$ computes

$$\begin{aligned}[\![w]\!][\![x]\!]^{-2r_x}[\![-r_x^2]\!] &= [\![w]\!][\![-2xr_x]\!][\![-r_x^2]\!] = \\ &= [\![w - 2xr_x - r_x^2]\!] = \\ &= [\![\underbrace{x^2 + 2xr_x + r_x^2}_{w} - 2xr_x - r_x^2]\!] = [\![x^2]\!] \end{aligned} \tag{5}$$

obtaining the square value of x encrypted. The protocol EncSquare requires the same number of rounds of EncMul but only $2 \times 2T$ bits (2 ciphertexts) are transmitted. Even the computational complexity is reduced to a total asymptotic number of 5 `expo` operations.

- BitMin protocol: the protocol BitMin is a widely used building block that computes the minimum between two encrypted values. In Fig. 4 the flow of the protocol is depicted.

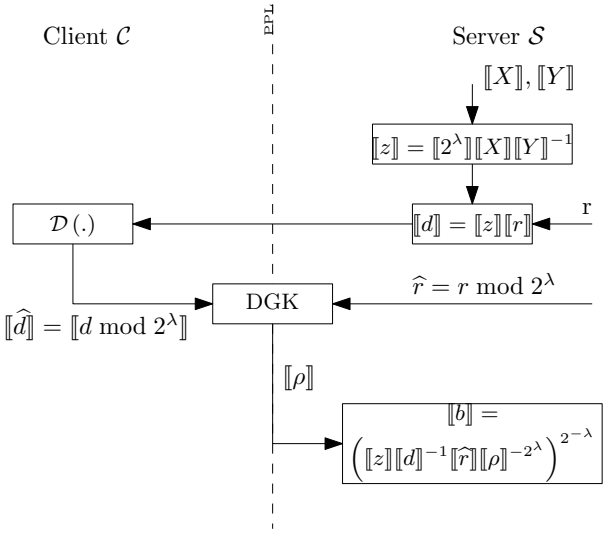


Fig. 4 The Protocol BitMin.

Given two encrypted values $[\![x]\!]$, $[\![y]\!]$, where x and y are ℓ-bit long, the main idea is to obtain the encryption of a bit b that assumes the value 0 if $x < y$, 1 otherwise. This can be done by computing the difference between the two values and extracting the sign bit. Even if this is a very simple operation when computed on plain values, it is not trivial when the values are encrypted. In the BitMin, $\mathcal{S}$ starts by computing $[\![z]\!] = [\![2^\ell + x - y]\!]$ by relying on homomorphic encryption, obtaining an $\ell + 1$-bit integer. The most significant bit of z (which we denote z_ℓ) is 0 if and only if $x < y$. Computing z_ℓ can be done as follows. $\mathcal{S}$ additively blinds z with a suitable random value r, obtaining $[\![d]\!]$, then it sends $[\![d]\!]$ to $\mathcal{C}$. At

this point $\mathcal{S}$ and $\mathcal{C}$ run a comparison protocol [24] after which $\mathcal{S}$ will learn $[\![\rho]\!]$ such that $\rho = 0 \Leftrightarrow [\hat{d} < \hat{r}] = [d \bmod 2^\ell < r \bmod 2^\ell]$. We notice that given ρ it is possible to compute z_ℓ as:

$$b = z_\ell = 2^{-\ell}(z - \hat{z}) = 2^{-\ell}(z - ((d-r) \bmod 2^\ell)), \tag{6}$$

where $(d-r) \bmod 2^\ell = (d \bmod 2^\ell) - (r \bmod 2^\ell) + \rho \cdot 2^\ell$.
The DGK comparison protocol allows both parties (i.e. $\mathcal{C}$ and $\mathcal{S}$) to learn the bit ρ of the predicate $d < r$, where d and r are two ℓ-bit integers owned by $\mathcal{C}$ and $\mathcal{S}$ respectively, by decomposing encrypted values into the encryptions of the single bits and returning the encryption of the most significant bit. For sake of brevity we do not describe the DGK protocol, interested readers may directly refer to [24].

Table 5 Computational Complexities – DGK sub-protocol.

#exp	Bandwidth	Rounds
4ℓ	$\frac{2\ell T}{3} + 1$	3

Considering the DGK protocol complexities shown in Tab. 5, the BitMin requires a number of rounds equal to 4: 1 to exchange the result and 3 due to DGK protocol. Only 1 ciphertext is sent from $\mathcal{S}$ to $\mathcal{C}$, so the bandwidth is a Paillier ciphertext plus the bandwidth of DGK, thus: $2T + \frac{2\ell T}{3} + 1 = 2T(1 + \frac{\ell}{3}) + 1$. Finally the asymptotic computation complexity is 1 `expo` to compute $[\![d]\!]$, $1\ \texttt{dec} + 1\ \texttt{enc}$ to obtain $[\![\widehat{d}]\!]$ and 3 `expo` to compute $[\![b]\!]$; that is 6 `expo`. Considering that DGK requires 4ℓ exponentiations we have: $(6 + 4\ell)$ `expo`.

3.2.3 Composite signal representation

A problem with the use of homomorphic encryption is that signals need to be encrypted sample-wise. Samplewise encryption of signals poses some severe complexity problems since it introduces a huge expansion factor between the original signal sample and the encrypted one.

To fix the ideas, let us assume that the Paillier cryptosystem is used; in this case each encrypted sample is an element of $\mathbb{Z}_{N^2}$, i.e. the set of integer numbers modulo N^2 with N being at least 1024 bit long, that is each encrypted sample needs at least 2048 bits to be represented. By considering that plain signal samples are usually represented by a few bits (e.g. 8 bits for images or 16 bits for ECG signals), we conclude that due to encryption, signals are expanded by a factor ranging from 125 to 250. For instance, the size of a grey level 1000×1000 image will pass from 1Mbyte in the clear to 250 Mbytes in the encrypted domain. This huge expansion factor is clearly not affordable in many practical applications.

In order to solve these problems, in [14] an alternative representation of signals has been proposed. This representation permits to greatly reduce the expansion factor introduced by encryption, while still allowing the exploitation of the homomorphic properties of the underlying cryptosystem to process signals in the encrypted domain. In addition to limiting the storage requirement, this representation allows the parallel processing of different samples, thus providing a considerable reduction of computational complexity in terms of operations between encrypted messages.

The main idea behind the representation is to pad multiple data samples to form a composite encrypted message. To be specific, let $\mathcal{M}$ be the message space and $\mathcal{C}$ the cypher space and let signal samples be l-bit long. It is possible to bundle R l-bit messages $m_1,\ldots,m_R$ within a single composite message x as follows:

$$x = m_1 \cdot 2^0 + m_2 \cdot 2^L + \ldots + m_R \cdot 2^{L(R-1)}. \tag{7}$$

If L is larger than l, samples will remain distinct in the composite representation; moreover, if L is sufficiently large, adding two composite messages will result in the addition of the single messages composing them, and multiplying the composite message by a constant factor, will be equivalent to multiplying each single message by the same factor. In [14] other ways to pack more messages together that permit more complex operations, such as linear filtering, have been proposed.

3.3 Garbled Circuits

Yao demonstrated in [69] that any function can be securely evaluated in a constant number of rounds and polynomial communication and computation overhead, proposing the garbled circuit (GC) protocol. While Yao's protocol has been thought to be of theoretical interest only for a long time, recent works have shown its efficiency [49, 46, 60] and usability by compilers for automatic generation of GC-based STPC protocols [51, 59].

Yao's Garbled Circuit approach [68] is one of the most efficient methods for secure evaluation of a boolean circuit C. To describe garbled circuits in a few words, we can say that Yao's idea is to encrypt (or garble) the nodes and the transitions of a boolean circuit such that who evaluates it may follow only a single evaluation path, defined by the circuit and the input attribute vector. Given a public boolean function $y = f(\mathbf{x}_\mathcal{C}, \mathbf{x}_\mathcal{S})$, where $\mathbf{x}_\mathcal{C}$ is the set of (binary) inputs belonging to $\mathcal{C}$ and $\mathbf{x}_\mathcal{S}$ those to $\mathcal{S}$, it is possible to represent $f(\cdot)$ by a boolean circuit C. $\mathcal{C}$ and $\mathcal{S}$ are interested to evaluate the circuit, without disclosing their inputs. At the end of the protocol the output will be available to $\mathcal{C}$ and optionally to $\mathcal{S}$.

The circuit, together with $\mathbf{x}_\mathcal{C}$ and $\mathbf{x}_\mathcal{S}$, is an input of a generic GC scheme, where one party ($\mathcal{S}$) constructs the circuit, then discloses the secrets necessary for the evaluation to the other party ($\mathcal{C}$) and $\mathcal{C}$ uses them to evaluate the circuit.

A garbled circuit (Fig. 5) can be associated to any function described by a boolean circuit and is composed by the following blocks:

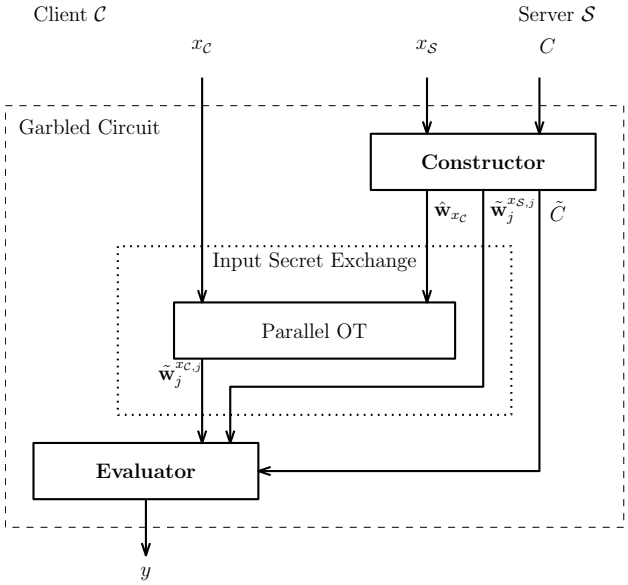


Fig. 5 General scheme for Garbled Circuits.

- **Constructor**: The circuit constructor, on $\mathcal{S}$ side, creates a *garbled circuit* $\widetilde{C}$:
  - for each input, intermediate and output wire W_i of the circuit, the constructor randomly chooses a complementary garbled value $\widehat{w}_i = \langle \widetilde{w}_i^0, \widetilde{w}_i^1 \rangle$ consisting of two secrets, $\widetilde{w}_i^0$ and $\widetilde{w}_i^1$, where $\widetilde{w}_i^j$ is the garbled value of W_i's value j, i.e. $\widetilde{w}_i^j$ is a randomly chosen secret associated to j that does not reveal j;
  - for each gate G_i, $\mathcal{S}$ creates a *garbled table* $\widetilde{T}_i$ with the following property: given a set of garbled values of G_i's inputs, $\widetilde{T}_i$ allows to recover the garbled value of the corresponding G_i's output, and nothing else.

  Each secret is randomly chosen and is uniformly distributed in the interval $(0, 2^t)$ (normally $t = 80$). Once the secrets are generated, for each gate, given the secrets $\widehat{w}_i$ and $\widehat{w}_j$ associated to the gate inputs wires and the secret $\widehat{w}_o$ associated to the gate output wire, the corresponding $\widetilde{T}$ is generated in the following way:

$$\begin{bmatrix} \mathcal{E}nc_{\widetilde{w}_i^0,\widetilde{w}_j^0}(\widetilde{w}_o^{g(0,0)}) \\ \mathcal{E}nc_{\widetilde{w}_i^0,\widetilde{w}_j^1}(\widetilde{w}_o^{g(0,1)}) \\ \mathcal{E}nc_{\widetilde{w}_i^1,\widetilde{w}_j^0}(\widetilde{w}_o^{g(1,0)}) \\ \mathcal{E}nc_{\widetilde{w}_i^1,\widetilde{w}_j^1}(\widetilde{w}_o^{g(1,1)}) \end{bmatrix} \tag{8}$$

  where $g(b_i, b_j)$ is the output of the gate. As to symmetric encryption, any algorithm having the following properties can be used:

  - *elusive range*: an encryption under one key is in the range of an encryption with a different key with negligible probability;
  - *efficiently verifiable range*: given a key, a user can efficiently verify that a ciphertext is in the range of that key.

  The rows of the tables are finally randomly scrambled to avoid that the evaluator understands the input values by the row successfully decrypted.
- **Input Secret Exchange**: Garbled values corresponding to $\mathcal{C}$'s inputs x_j are (obliviously) transferred to $\mathcal{C}$ with a parallel oblivious transfer (OT) protocol. An OT protocol [29] is a STPC tool where one party ($\mathcal{S}$) inputs two messages m_0 and m_1, while the other party ($\mathcal{C}$) inputs a bit b; at the end of the protocol $\mathcal{C}$ obtains the message m_b while nothing is revealed to $\mathcal{S}$. In the parallel OT inside the GC protocol, $\mathcal{S}$ inputs complementary garbled values $\hat{w}_j$, while $\mathcal{C}$ inputs $x_{\mathcal{C},j}$ and obtains $\widetilde{w}_j^{x_{\mathcal{C},j}}$ as outputs. Oblivious Transfer can be instantiated efficiently as shown in [54, 2], and by relying on elliptic curve cryptography. In addition, as shown in [11], the OTs can be pre-computed in a setup phase, such that they are not the performance bottleneck in Yao's protocol. Finally, the number of computationally expensive public-key operations in the setup phase can be reduced to a constant number with the extensions proposed in [40]. By considering these instantiations, a parallel OT of n secrets each t-bit long requires 2 rounds where $2nt$ bits are transmitted. After the OT, $\mathcal{S}$ transmits the secrets $\widetilde{w}_j^{x_{\mathcal{S},j}}$ relative to its input $x_{\mathcal{S},j}$ and the tables $\widetilde{T}_i$ of the circuit.
- **Evaluator**: $\mathcal{C}$ simply evaluates the garbled circuit $\widetilde{C}$ gate by gate, using the garbled tables $\widetilde{T}_i$, to obtain the garbled output. In each table the evaluator decrypts each row by using the input secrets previously obtained until it successfully performs a decryption. In the first gates only input secrets are used, while successively input secrets and/or secrets obtained as output from other tables are used. Finally, $\mathcal{C}$ determines the plain values corresponding to the obtained garbled output values using an output translation table received by $\mathcal{S}$. If the output is needed by $\mathcal{S}$, $\mathcal{C}$ transmits the garbled output.

The basic GC protocol outlined above can be improved in many ways as shown in [51] and [46]. In particular, in [51] the authors suggest to replace encryption by Hash functions and the scheme proposed in [46] allows "free" evaluation of XOR gates so that a garbled XOR gate has no garbled table and its evaluation consists of XOR-ing its garbled input values, resulting in *no communication* and *negligible computation*.

From a computational point of view, GCs have lower complexity than homomorphic encryption protocols, replacing exponentiations computed on large numbers with simple hash functions. On the other side the amount of transmitted data can grow quickly, considering that, given the number of bits ℓ necessary to represent each input value, we have to transmit the secrets relative to the bits of the $\mathcal{S}$ inputs ($\mathcal{O}(\ell)$), the data necessary for the parallel OT that returns the secrets relative to $\mathcal{C}$ inputs ($\mathcal{O}(\ell)$) and the garbled table ($\mathcal{O}(f(\ell))$), where $f(\cdot)$ depends on the particular functionality implemented by the circuit. Finally the number of rounds is the

same for any circuit (2) and it does not change if we assemble many building blocks together.

3.3.1 Basic building blocks

Many basic building blocks can be built by relying on GC theory. Fig. 6 and Fig. 7 show the circuits implementing the blocks that we will use later in Sect. 4 to build the primitives necessary to construct pattern matching protocols working on encrypted data. Being the figures self–explicative, we remind to the original papers for their detailed descriptions.

In the following we describe the implementations for the product and square protocols that are only outlined in [44]).

- **Product** MULT_ℓ [44]: to multiply two unsigned integers x and y represented with ℓ bits, we can construct a circuit according to the scholar method for multiplication, i.e., adding up the bitwise multiplications (logical AND) of y_i and x left shifted of i positions: $x \cdot y = \sum_{i=0}^{\ell-1} (y_i \wedge x) 2^i$. The circuit is composed of ℓ^2 AND gates (Fig. 7(e)) yielding the matrix

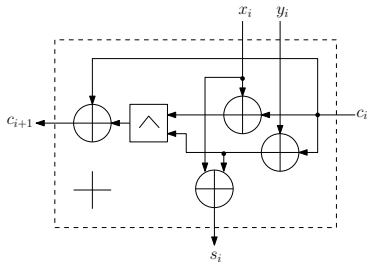
$$\begin{array}{cccc} y_0x_{\ell-1} & \cdots & y_0x_1 & y_0x_0 \\ y_1x_{\ell-1} & \cdots & y_1x_1 & y_1x_0 \\ & & \cdots & \\ y_{\ell-1}x_{\ell-1} & \cdots & y_{\ell-1}x_1 & y_{\ell-1}x_0 \end{array} \tag{9}$$

  and $\ell - 1$ adders (Fig. 7(f)). Instead of using adders of 2ℓ bits we can set $(x \cdot y)_0 = x_0 \wedge y_0$ and then add $(y_0 \cdot x)_{\ell-1,\ldots,1}$ with $y_1 \cdot x$ setting $(x \cdot y)_1$ equal to the least significant bit of the result and then adding the other bits to $x_2 \cdot y$, etc. In this way adders of values represented with ℓ bits are used. The circuit requires $\ell^2 + (\ell-1)\ell = 2\ell^2 - \ell$ non-XOR gates.
- **Square** SQUARE_ℓ: a circuit computing the square of an unsigned integer x can be obtained by optimizing the product circuit, that is replacing the circuit of Fig. 7(e) with the circuit of Fig. 7(g). By considering that $x_i \wedge x_i = x_i$ and $x_i \wedge x_j = x_j \wedge x_i$ we can rebuild the matrix of (9) as:

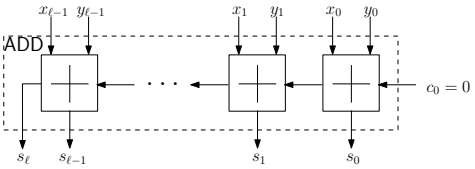
$$\begin{array}{cccc} x_0x_{\ell-1} & \cdots & x_0x_1 & x_0 \\ x_1x_{\ell-1} & \cdots & x_1 & x_0x_1 \\ & & \cdots & \\ x_{\ell-1} & \cdots & x_1x_{\ell-1} & x_0x_{\ell-1} \end{array} \tag{10}$$

  In this way only $\ell - 1 + \ell - 2 + \cdots + 1 = \ell(\ell-1)/2$ AND gates are evaluated, obtaining $\frac{3}{2}\ell(\ell-1)$ gates for the whole circuit.

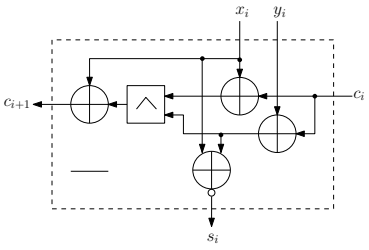
Reminding that the circuit complexity is related to the number of non-XOR gates (each having a table of size $4t$ bits associated), Table 6 shows the complexity of the circuits as a function of the number of non-XOR gates.



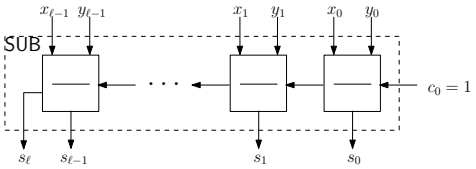
(a) 1-bit adder



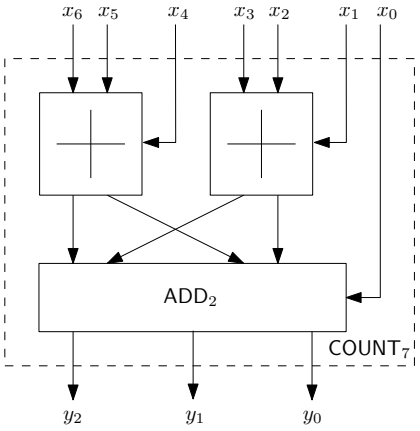
(b) ADD_ℓ circuit implemented by using sub-circuit of Fig. 6(a)



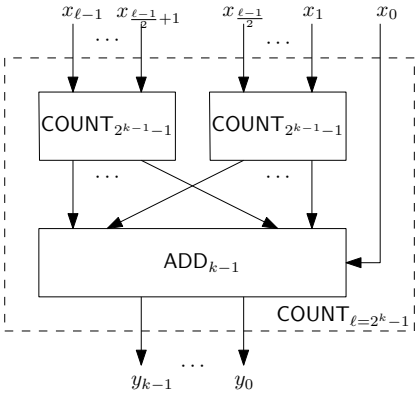
(c) 1-bit subtractor



(d) SUB_ℓ circuit implemented by using sub-circuit of Fig. 6(c)

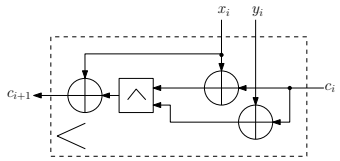


(e) COUNT_7

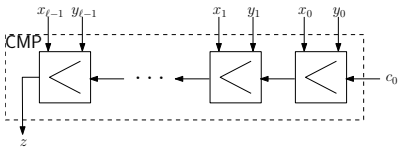


(f) $\mathsf{COUNT}_{\ell=2^k-1}$ recursively developed with smaller $\mathsf{COUNT}_{\ell=2^{k-1}-1}$. Generic COUNT_ℓ can be obtained by optimizing greater $[\mathsf{COUNT}_{2^k-1}$.

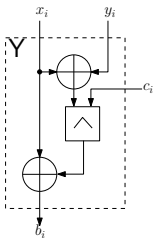
Fig. 6 Logical circuits implementing the basic building blocks used in Sect. 4 (first part).



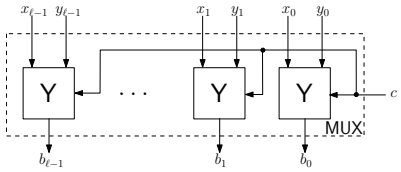
(a) 1-bit comparison



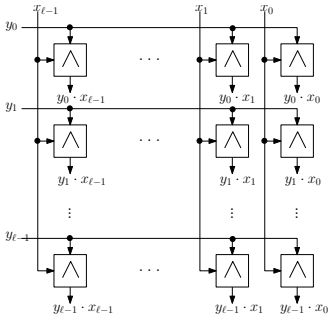
(b) Comparison circuit circuit implemented by using sub-circuit of Fig. 7(a)



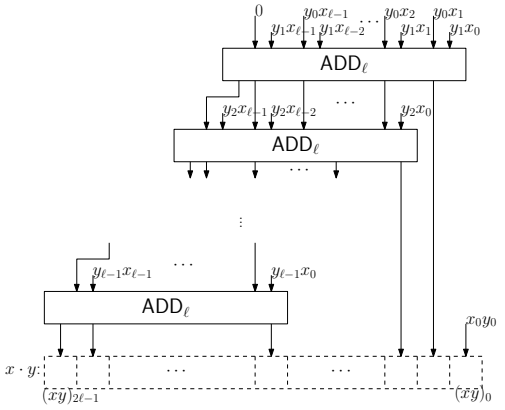
(c) Y block



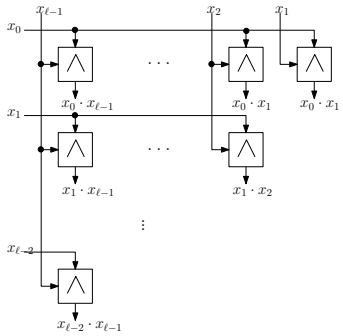
(d) MUX_ℓ circuit implemented by using sub-circuit of Fig. 7(c)



(e) First part of MULT_ℓ



(f) Second part of MULT_ℓ



(g) First part of SQUARE_ℓ (The second part is similar to Fig. 7(f))

Fig. 7 Logical circuits implementing the basic building blocks used in Sect. 4 (second part).

Table 6 Complexity of GCs implementing the basic building blocks used in Sect. 4.

Circuit	Inputs (bit length)	Output (bitlength)	# non-XOR gates
ADD_ℓ [44], Fig. 6(b)	$x\ (\ell)$, $y\ (\ell)$	$s\ (\ell+1)$	ℓ
SUB_ℓ [44], Fig. 6(d)	$x\ (\ell)$, $y\ (\ell)$	$s\ (\ell+1)$	ℓ
COUNT_ℓ [10], Fig. 6(f)	$x\ (\ell)$	$y\ (k \approx \log_2(\ell+1))$	$\approx \ell - k$
$<_\ell$ [44], Fig. 6(d)	$x\ (\ell)$, $y\ (\ell)$	$z\ (1)$	ℓ
MUX_ℓ [46], Fig. 7(d)	$x\ (\ell)$, $y\ (\ell)$, $c\ (1)$	$b\ (\ell)$	ℓ
MULT_ℓ [44], Fig. 7(e),7(f)	$x\ (\ell)$, $y\ (\ell)$	$z\ (2\ell)$	$2\ell^2 - \ell$
SQUARE_ℓ, Fig. 7(g)	$x\ (\ell)$	$z\ (2\ell)$	$\frac{3}{2}\ell^2 - \frac{3}{2}\ell$

3.4 Hybrid protocols

Given STPC protocols implementing several basic functions it is possible to obtain more complicated protocols by composing them. Homomorphic Encryption is particularly useful when it is possible to move the computation on $\mathcal{S}$'s side (almost) without interaction, while Garbled Circuits are more performing when the data is represented by few bits or whenever it is not possible to perform some operations by HE. As a result, it is possible that complex protocols contain blocks having an efficient HE implementation, while others can be more efficiently implemented by using GCs. To pass from HE to GC and viceversa it is necessary to disclose intermediate values to $\mathcal{C}$ (that owns the decryption key of the homomorphic cryptosystem), but this involves a privacy leakage. To solve this problem we can use blinding [45]: the intermediate data is first of all blinded by adding a random value (known only by $\mathcal{S}$) and then disclosed to $\mathcal{C}$. The following HE or GC sub-protocol will remove the obfuscation before continuing the computation.

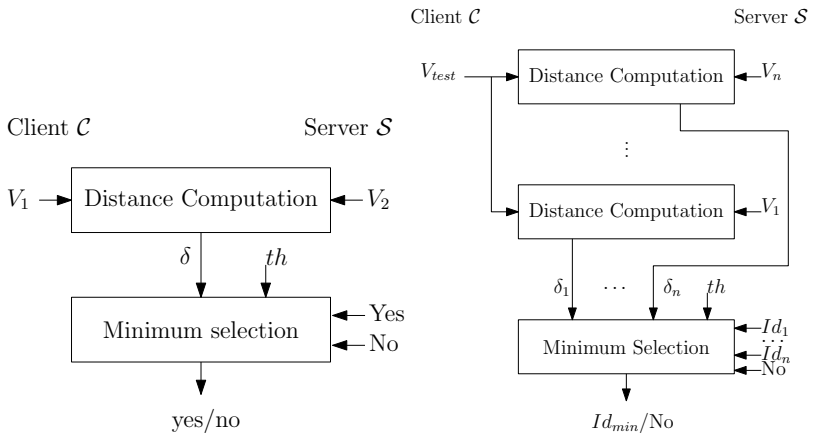
For example, to convert an Homomorphic value $[\![x]\!]$ into a Garbled value $\widetilde{x}$, $\mathcal{S}$ adds a random value r under homomorphic encryption, sends the blinded value $[\![\bar{x}]\!] = [\![x+r]\!] = [\![x]\!] \cdot [\![r]\!]$ to $\mathcal{C}$ who decrypts it and uses the $\bar{x}$ value as input to the subsequenting GC. $\mathcal{S}$ inputs the value r to the GC and the constructor will prepare a garbled circuit that first computes the subtraction between $\bar{x}$ and r and then evaluates the desired block. A similar method can be used for converting a Garbled value $\widetilde{x}$ into a Homomorphic value $[\![x]\!]$.

4 Building blocks for privacy-aware pattern matching

We now describe the building blocks necessary to carry out the general secure pattern matching algorithms described in Sect. 2.

The first problem to be considered is the verification problem. Pattern verification boils down to the computation of a certain distance function and its comparison against a threshold, as shown in Fig. 8(a). It is easy to realize that the verification problem can be considered as special case of the identification problem, shown in

Fig.8(b). Verification only needs that the distance between V_1 (now playing the role of V_{test}) and V_2 is computed and the minimum between such a distance and the verification threshold evaluated. So, in the following we will treat the two problems together.



(a) Verification problem (b) Identification problem

Fig. 8 STPC blocks necessary for the Matching problems

Specifically we will describe the single blocks necessary for secure evaluation of the Matching Problems: *distance computation* and *minimum selection*. For each block the implementation by using HE and GC will be provided. The comparison among the different implementations and their composition will be then analyzed in Sect. 5.

4.1 Distance Computation

This Section is devoted to sketch the sub-protocols for securely computing the distance δ between two feature vectors. We will describe two possible solutions: the former relying on homomorphic encryption, the latter on garbled circuits. Due to the great difference in complexities and performances of these approaches, we will provide an analysis of our constructions in the following Sect. 5, trying to delineate the different contexts in which one approach should be preferred to the other. Considering the verification problem the distance δ is computed between V_1 and V_2, while in the identification problem n distances δ_i are evaluated between the feature vector V_{test} provided by $\mathcal{C}$ and the feature vectors V_i stored in the database owned by $\mathcal{S}$. Reminding that each biometry feature vector can be represented by a point in $\mathbb{Z}_b^m$, as presented in Sect. 2, for sake of notation simplicity the distance computation

is here evaluated between two points $P,Q \in \mathbb{Z}_b^m$ and the final result belongs to $\mathbb{Z}_{m'}$, where m' is chosen to correctly represent the distance.

4.1.1 Euclidean Distance

The Euclidean distance is defined as

$$\delta = d_E(P,Q) := \sqrt{\sum_{j=1}^{m}(p_j - q_j)^2}$$

for $j = 1,\ldots,m : p_j, q_j \in [0,\ldots,b) \subseteq \{0,1\}^\ell$. As the Euclidean distance is non-negative, in many cases it is replaced by $\delta^2 = \sum_{j=1}^{m}(p_j - q_j)^2$, namely the square of the Euclidean distance, whose minimum coincides with the minimum among Euclidean distances. Considering that $P,Q \in \mathbb{Z}_b^m$, we can observe easily that $\delta^2 \in \mathbb{Z}_{m'}$, where $m' = m * (2b)^2$.

Homomorphic Protocol

The encryption of the square of the Euclidean distance $[\![\delta_i^2]\!] = [\![d_E(P_i,Q)^2]\!]$ can be computed by using additively homomorphic encryption together with an additional round for squaring as proposed in [28]. As depicted in Fig. 9, $\mathcal{S}$ is able to compute all the differences $[\![p_j - q_j]\!]$ by using the additive homomorphic property, then the interactive EncSquare protocol is needed to compute the squared values of the summands, to let $\mathcal{S}$ obtain $[\![(p_j - q_j)^2]\!]$.

Considering that many calls to EncSquare are required, they can be evaluated in parallel. In this way, with just two rounds $2mn$ cyphertexts of size $2T$ bits are exchanged between $\mathcal{C}$ and $\mathcal{S}$ ($n > 1$ when more distances are parallel evaluated).

Finally by the homomorphic properties $\mathcal{S}$ can compute δ^2 by multiplying in the encrypted domain (equivalent to adding in the plain domain) all the squared values.

GC protocol

To build a GC-based STPC for computing the euclidean distance, we need to pay particular attention to the correct number of bits used to represent each value involved in the computation. In this case a Boolean circuit computing $\sum_{j=1}^{m}(p_j - q_j)^2$ is evaluated. Supposing that each feature of the biometry is represented with ℓ bits (the base used for the feature representation is $2^{\ell-1} \leq b < 2^\ell$), the point P and Q are represented with $m\ell$ bits each. The differences between each couple of features are represented with $\ell + 1$ bits and the square values of the difference needs $2\ell + 2$ bits for their representation. Finally δ is obtained by adding all the square differences and is represented with $2\ell + 2 + \lceil \log_2 m \rceil$ bits.

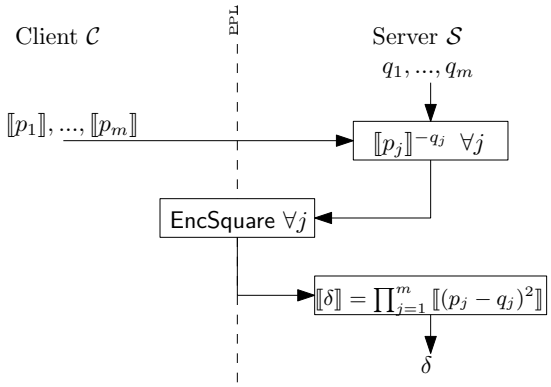


Fig. 9 Euclidean Distance via Homomorphic Encryption.

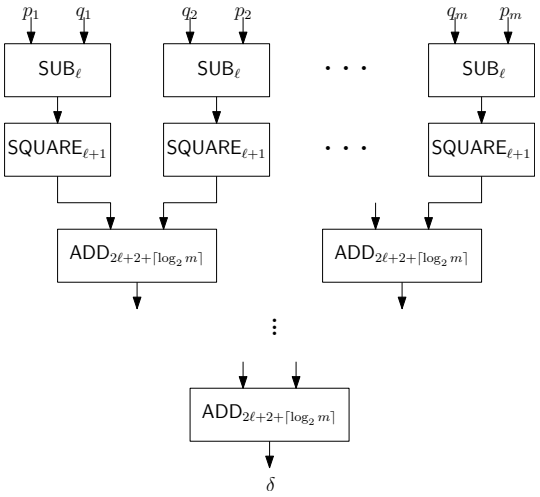


Fig. 10 Euclidean Distance via Garbled Circuits.

The circuit, shown in Fig. 10, requires m SUB_ℓ, m $\mathsf{SQUARE}_{\ell+1}$ and $(m-1)$ $\mathsf{ADD}_{2\ell+2+\lceil \log_2 m \rceil}$.

To transmit the circuit $(m\ell + \frac{3m}{2}(\ell+1)\ell + (m-1)(2\ell+2+\lceil \log_2 m \rceil))4t$ bits are transferred from $\mathcal{S}$ to $\mathcal{C}$.

4.1.2 Hamming Distance

The Hamming distance is often used when biometries are represented by vectors of boolean features (i.e. points $P, Q \in \mathbb{Z}_2^m$), and is defined as $d_H(P,Q) := \sum_{j=1}^{m} p_j \oplus q_j \in \mathbb{Z}_{m'}$, where $m' = \lceil \log_2 m \rceil$.

Homomorphic Protocol

To evaluate the Hamming distance in a privacy preserving fashion, the m XOR operators needed for the distance evaluation are computed by using homomorphic encryption. Let us assume that we want to evaluate a generic $[\![p_j \oplus q_j]\!]$ where p_j and q_j are bit values available in encrypted format, i.e. $\mathcal{S}$ knows $[\![p_j]\!]$ and $[\![q_j]\!]$, where encryption is carried out by using $\mathcal{C}$'s *PuK*. In this setting $\mathcal{S}$ does not want to reveal neither p_j nor q_j to $\mathcal{C}$, so it chooses two additional random bits r_{p_j} and r_{q_j} and computes $[\![p_j \oplus r_{p_j}]\!] = [\![p_j + r_{p_j} - 2p_j r_{p_j}]\!] = [\![p_j]\!]^{1-2r_{p_j}} [\![r_{p_j}]\!]$ and similarly $[\![q_j \oplus r_{q_j}]\!]$ then it sends these values to $\mathcal{C}$. The obfuscated bits can be packed in a single cyphertext by computing $[\![p_j \oplus r_{p_j}]\!]^2 [\![q_j \oplus r_{q_j}]\!] = [\![2(p_j \oplus r_{p_j}) + (p_j \oplus r_{p_j})]\!]$. Note that p_j and q_j are perfectly obfuscated by the xor-ing with r_{p_j} and r_{q_j}, so $\mathcal{C}$ can safely decrypt the cyphertexts, obtain the single bits, compute the encryption of $[\![(p_j \oplus r_{p_j}) \oplus (q_j \oplus r_{q_j})]\!]$ and send the result back to $\mathcal{S}$. At this point $\mathcal{S}$ can remove $r_{p_j} \oplus r_{q_j}$ from the result and obtain $[\![p_j \oplus q_j]\!]$. The whole protocol is shown in Fig. 11.

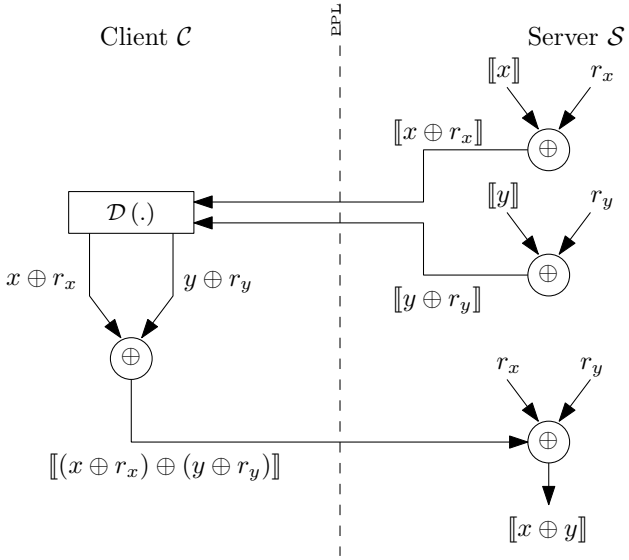


Fig. 11 Sub protocol XOR with $[\![x]\!]$ and $[\![y]\!]$.

Since $\mathcal{S}$ needs to compute the XOR function, $\mathcal{C}$ computes two decryptions and one encryption and the complexity is $4\,\texttt{expo} + 2\,\texttt{dec} + 1\,\texttt{enc} \simeq 7\,\texttt{expo}$.

Since computing the Hamming distance requires m XOR operations, the communication complexity can be reduced by packing all the nm $p_j \oplus r_{p_j}$ ($n > 1$ when several distances are parallel computed) and all the nm $q_j \oplus r_{q_j}$ in a single cyphertext during the transmission from $\mathcal{S}$ to $\mathcal{C}$ answers with nm cyphertexts. The round complexity is 2.

GC protocol

In Hamming distance calculation each feature is represented by 1 bit, hence the points P and Q are represented with m bits. The XOR among the two points is again represented with m bits, while the distance can be represented with $\lceil \log_2(m) \rceil$ bits. The circuit computing the Hamming distance is composed by m XOR (having no tables associated thanks to the Free-XOR) and their binary results are summed together by using a COUNT_m having $\lessapprox m - \log_2(m+1)$ non-XOR gates. The garbled gates transmitted are hence relative only to the COUNT circuit, implying the transmission of less than $(m - \log_2(m+1))4t$ bits.

4.2 Minimum Selection

The computation of the minimum among a set of values is the second essential operation needed in a matching protocol. When the minimum has to be computed among $n+1$ values, and the index of the minimum value is required, we can use the cascade of several minimum blocks as depicted in Fig. 12.

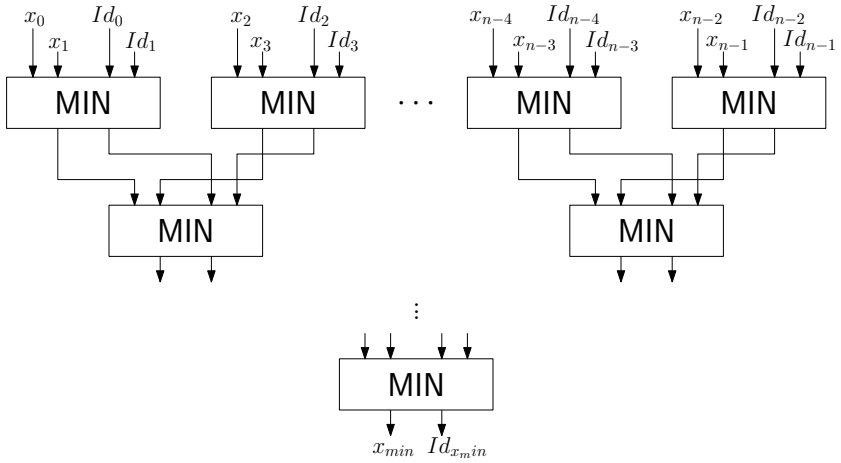


Fig. 12 Circuit for Minimum search among n values

Due to this we can solve the problem of Minimum Selection by repeatedly application of a basic building block able to compute the minimum and the related index on a couple of values.

Suppose we have two integer values x, y represented with ℓ bits obtained from a previous computation and two identification labels Id_x, Id_y associated to x and y respectively and represented with κ bits. The goal of Minimum Selection is to select $\min\{x, y\}$ and the $Id_{\{x,y\}}$ associated to the minimum.

Homomorphic protocol

Given the sub-protocol BitMin described in Sect. 3.2.2, allowing $\mathcal{S}$ to compute the encrypted bit $[\![b]\!]$ such that:

$$b = \begin{cases} 0 \text{ if } x < y \\ 1 \text{ if } x \geq y, \end{cases}, \tag{11}$$

$\mathcal{S}$ can compute $[\![\min\{x,y\}]\!] = \mathsf{EncMul}([\![1-b]\!], [\![x]\!][\![y]\!]^{-1})[\![y]\!] = [\![(1-b)(x-y)+y]\!]$ and $[\![Id_{\min\{x,y\}}]\!] = \mathsf{EncMul}([\![1-b]\!], [\![Id_x]\!][\![Id_y]\!]^{-1})[\![Id_y]\!] = [\![(1-b)(Id_x - Id_y)+Id_y]\!]$. Note that the two EncMul can be performed in parallel and b is transmitted only once, resulting in the transmission of 5 cyphertexts.

When the minimum among $n+1$ values is evaluated returning its index, as in the identification problem where there is the necessity to choose among n distances and a threshold, we need to evaluate n minimum functions as shown in Fig. 12. For each minimum block $2T(6+\frac{\ell}{3})+1$ bits are transmitted and $14+4\ell$ `expo` are evaluated during the computation. The reverse tree has $\lceil \log_2(n+1) \rceil$ levels and 6 rounds are required for each level.

GC protocol

The minimum circuit (MIN) which selects the minimum value $\min\{x,y\}$ among two values x and y together with the associated Id is shown in Fig. 13. The circuit is

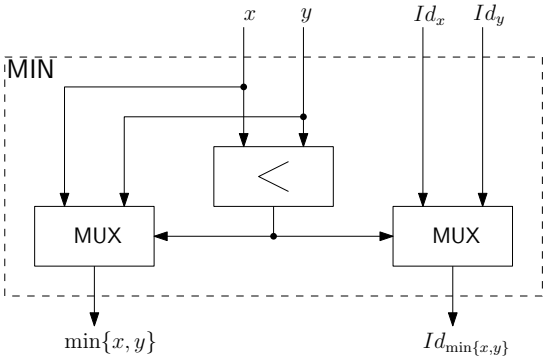


Fig. 13 The circuit which selects the minimum value $\min\{x,y\}$ among x and y together with the Id associated.

composed by a comparison circuit having ℓ non-XOR gates and two MUX circuits. The one that selects among x, y has ℓ non-XOR gates, while that selecting among the Ids has κ non-XOR gates, totally resulting in $2\ell+\kappa$ tables of size $4t$ bits each. If only the Id of the minimum value is required the MUX selecting among x and y has to be removed, resulting in a total size of $\ell+\kappa$ gates.

When the minimum has to be computed among more than two values we can use a reverse tree structure, as in Fig. 12. The minimum value and minimum identifier are selected pairwise in a tournament-like way using a reverse tree of minimum blocks, were the intermediate blocks choose among the outputs of the previous blocks. Given $n+1$ pairs (x_i, Id_i) the circuit needs n MIN block, hence the total size of the garbled circuit is $n(2\ell+\kappa)$ tables of $4t$ bits. If only the Id of the minimum value is required, in the last MIN block the MUX selecting among the values can be removed, resulting in a total size of $n(2\ell+\kappa)-\ell$ gates.

5 Design principles for privacy-aware biometric matching

By composing the building blocks described in Sect. 4, a privacy-aware matching protocol can be easily built. In this Section we give some general guidelines that can be used to choose the proper implementation of the basic building blocks so to achieve an efficient protocol suited to the application at hand.

From a computational point of view, HE is preferrable when the parties have enough computing power since they need to compute many exponentiations, while GC requires the computation of (many) simple Hash functions. From a communication point of view HE protocols transmit few long cyphertexts ($2T$ bits each, where T is at least 1024) in a number of rounds that depends on the application, while GC has to transfer a secret for any $\mathcal{S}$ input bit ($t = 80$ bits long each), a table for each non-XOR gate ($4t$ bits) and exchange the secrets relative to the $\mathcal{C}$ input bits by OT ($2t$ bits transferred for each input bit), resulting in a high bandwidth, even if all the transmissions are performed in only 2 rounds.

Comparing the HE and the GC implementations of the subprotocol (detailed in Sect. 4) from a computational point of view is not easy, since the answer finally depends on the number of bits used to represented biometric vectors and hence on the representation accuracy needed to achieve a given recognition rate. Usually a HE protocol is composed by few difficult operations performed on large numbers; in a Paillier homomorphic protocol, where cyphertexts are represented by $2T$ bits ($T = 1024$), the most complex basic operation is the exponentiation, having $\mathcal{O}(T^3)$ complexity. On the other side to evaluate a GC a big number of hash functions, having small constant complexity, are computed. Moreover we have to consider that the complexity of these operations can change from a system to an other, for example in the presence of dedicated hardware. Finally we have to consider that in the future longer cyphertexts will be required to guarantee security, due to the increase of available computational resources. While the security parameter T of an asymmetric scheme (necessary to HE) grows exponentially, the security parameter t of the symmetric cryptosystem (used in GC) grows linearly [34], making GC more performing than HE.

We now analyze the different implementations of the building blocks described in Sect. 4, from a communication point of view.

We remind that any matching protocol starts with the computation of distances (1 distance in the verification scenario and n distances in the identification scenario), and then terminates with a minimum selection, computed between two values (the distance and a threshold) in the verification protocol or $n+1$ values (the n distances and a threshold) in the identification protocol. A label represented with κ bits is associated to each distance (for example "Yes" in the verification protocol or an identifier in the identification protocol) and to the threshold (the "No" string).

Given a feature vector composed by m features represented with $\ell < T$ (usually $\ell << T$) bits each, if the euclidean distance is computed by HE, $\mathcal{C}$ transmits m cyphertexts to $\mathcal{S}$ and during the computation other $2m$ cyphertexts are transmitted for each distance (note that it is possible to reduce the transmission from $\mathcal{S}$ to $\mathcal{C}$ by packing several values in a single ciphertext), resulting in the transmission of $\mathcal{O}(nmT)$ bits in 3 rounds and the distances are finally available to $\mathcal{S}$ in encrypted form. In a GC protocol $\mathcal{C}$ has to obtain $m\ell$ secrets by the parallel OT ($2m\ell t$ bits); receives $nm\ell$ secrets relative to the feature vectors representing the biometries owned by $\mathcal{S}$ ($nm\ell t$ bits); receives $m\ell + \frac{3m}{2}(\ell+1)\ell + (m-1)(2\ell+2+\lceil \log_2 m \rceil)$ garbled tables for each distance computation ($n(m\ell + \frac{3m}{2}(\ell+1)\ell + (m-1)(2\ell + 2 + \lceil \log_2 m \rceil))4t$ bits), obtaining a total communication complexity of $\mathcal{O}(nm\ell^2 t)$ bits. Finally the output secrets are available to $\mathcal{C}$. We can observe easily that GC is preferable to HE only when ℓ is small.

When $\ell = 1$, Euclidean distance is replaced by Hamming distance. HE still requires the transmission of m cyphertexts from $\mathcal{C}$ to $\mathcal{S}$ at the beginning of the protocol, but then the communication complexity is reduced to $nm+1$ cyphertexts. The asymptotic communication complexity results $\mathcal{O}(nmT)$ bits. The GC computing the Hamming distance requires a parallel OT for m secrets ($2mt$ bits), the transmission of nm secrets (nmt bits) and $\lessapprox n(m - \log_2(m+1))$ non-XOR gates of size $4t$ bits, with an asymptotic communication complexity of $\mathcal{O}(nmt)$ bits. Concluding, when Hamming distance can be computed, GC is preferable to HE also from a communication complexity point of view.

Regarding the minimum selection GC is indeed more efficient than HE. In fact in a GC solution only two rounds are necessary (there are no additional rounds if the distance computation is also carried out by GC) and $\mathcal{O}(n\ell_d t)$ bits are transferred, where ℓ_d is the number of bits necessary to represent a distance, while the HE solution requires $6\log_2(n+1)$ rounds where $\mathcal{O}(n\ell_d T)$ bits are transmitted.

The asymptotic complexities are summarized in Tab. 7.

Table 7 Asymptotic communication complexities of the different implementation for the addressed subprotocols.

	Euclidean distance		Hamming distance		Minimum selection	
	Bandwidth	Rounds	Bandwidth	Rounds	Bandwidth	Rounds
HE	$\mathcal{O}(nmT)$	3	$\mathcal{O}(nmT)$	3	$\mathcal{O}(n\ell_d T)$	$6\log_2(n+1)$
GC	$\mathcal{O}(nm\ell^2 t)$	2	$\mathcal{O}(nmt)$	2	$\mathcal{O}(n\ell_d t)$	2

To conclude, for both verification and identification, we suggest to use a hybrid protocol where the distance is computed by HE and the minimum is selected by GC. When the biometries can be represented by binary vectors, we suggest to evaluate the Hamming distance by using GC, obtaining a unique GC that computes distances and the minimum index.

6 Conclusions

Multiparty computation has been studied for three decades by cryptographers, however only recently the state of the art in the field, and the computational power and bandwidth made available by information and communication technology have permitted to deploy such techniques for real life applications. Among the most promising applications of SMPC (and specifically STPC), privacy-aware processing of biometric data occupies a central role. As a matter of fact, biometric applications raise important privacy issues that can be conveniently solved by resorting to STPC. In this chapter we have reviewed the basic concepts behind STPC and described the basic cryptographic primitives needed to achieve privacy-aware processing of biometric data in a STPC context. The two main approaches proposed so far, namely homomorphic encryption and garbled circuits have been discussed and the way such techniques can be used to develop a full biometric matching protocol described. Some general rules designers should follow to select the most appropriate tools have also been given.

Even if the state of the art already permits the development of real-life applications based on the tools described in this chapter, several advances are still needed before we assist to a widespread use of STPC techniques in biometric applications.

The most pressing demand is surely a request for a better efficiency. The importance of this request lies in a very simple fact: privacy has a price and if we want that someone pays for improving the privacy of a system this price must be reasonably low. Actually it is surprising how few people are willing to pay for privacy measures despite the continuous call for privacy raising from various sources. While everybody agree that sensitive data need to be protected and that personal privacy is worth protection, very few users would be willing to pay for a secure service if an insecure, but faster, service is offered to them for free or at a lower price.

A second line of research that needs to be considered regards the security model. According to the current state of the art, efficient privacy preserving protocols are available only under the assumption of semi-honest parties. This is a rather common assumption, however its applicability in practical scenarios is doubtful. In most cases, in fact, we should assume that our adversary is willing to deviate from the correct protocol if in this way he can steal some supposed-to-be-secret information. Some interesting results in this sense have been shown in [55]. We expect that further improvements will follow hence making privacy protection in the presence of a malicious adversary practical.

Finally, we mention the importance that specific biometric processing algorithms are devised tailored to the need of a privacy-preserving implementation. Indeed, the approach used so far has been that of taking a classical algorithm and transforming it into a protocol to be run on encrypted signals. It is arguable that much better results could be obtained by developing a class of processing tools that are explicitly thought to ease a STPC implementations, e.g. by considering in advance which are the most complex operations to be carried out in a secure way and try to avoid them, or by trying to minimize the number of bits used to represent the biometric templates to reduce the communication or computational complexity of the protocols.

References

1. R. Agrawal and R. Srikant. Privacy-preserving data mining. *Sigmod Record – ACM*, 29(2):439–450, 2000.
2. B. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. *Advances in Cryptology - EUROCRYPT 2001*, pages 119–135, 2001.
3. E. Barker, W. Burr, A. Jones, T. Polk, S. Rose, M. Smid, and Q. Dang. Recommendation for Key Management. *NIST special publication*, 800:57, 2009.
4. W. C. Barker. *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*. NIST special publication, May 2004.
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 *Multimedia and security, Proceedings of the 12th ACM workshop on*, pages 231–240. ACM, 2010.
6. M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R.D. 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*, pages 1–7. IEEE, 2010.
7. M. Barni, P. Failla, V. Kolensikov, R. Lazzeretti, A. Paus, A. Sadeghi, and T. Schneider. Efficient Privacy-Preserving Classification of ECG Signals. In *Information Forensics and Security – WIFS 2009, Workshop on*, 2009.
8. M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.R. Sadeghi, and T. Schneider. Secure evaluation of private linear branching programs with medical applications. *Research in Computer Security – ESORICS 2009, European Symposium on*, pages 424–439, 2010.
9. M. Barni, P. Failla, R. Lazzeretti, A.R. Sadeghi, and T. Schneider. Privacy-Preserving ECG Classification with Branching Programs and Neural Networks. *Information Forensics and Security – TIFS, IEEE Transactions on*, Jun. 2011.
10. M. Barni, J. Guajardo, and R. Lazzeretti. Privacy preserving evaluation of signal quality with application to ECG analysis. In *Information Forensics and Security – WIFS 2010, IEEE International Workshop on*, pages 1–6. IEEE, 2010.
11. D. Beaver. Precomputing oblivious transfer. In *Advances in Cryptology – CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, 1995.
12. J. Benaloh. Dense probabilistic encryption. In *Selected Areas of Cryptography, Proceedings of the Workshop on*, pages 120–128. Citeseer, 1994.
13. T. Bianchi, A. Piva, and M. Barni. On the implementation of the discrete Fourier transform in the encrypted domain. *Information Forensics and Security, IEEE Transactions on*, 4(1):86–97, 2009.
14. T. Bianchi, A. Piva, and M. Barni. Composite signal representation for fast and storage-efficient processing of encrypted signals. *Information Forensics and Security, IEEE Transactions on*, 5(1):180–187, 2010.

15. A. Bjorck. Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1):1–21, 1967.
16. D. Boneh, E.J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. *Theory of Cryptography*, pages 325–341, 2005.
17. D. Boneh and R. Lipton. Algorithms for black-box fields and their application to cryptography. In *Advances in Cryptology - CRYPTO96*, pages 283–297. Springer, 1996.
18. E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. *Advances in Cryptology-ASIACRYPT 2003*, 1:37–54, 2003.
19. J. Brickell, D.E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving remote diagnostics. In *Computer and communications security, Proceedings of the 14th ACM conference on*, page 507. ACM, 2007.
20. R. Brinkman, J. Doumen, and W. Jonker. Using secret sharing for searching in encrypted data. *Secure Data Management*, 1:18–27, 2004.
21. JL Camp. Digital identity. *IEEE Technology and Society Magazine*, 23(3):34–41, 2004.
22. C.K. Chui and E. Quak. Wavelets on a bounded interval. *Numerical methods of approximation theory*, 9(1):53–57, 1992.
23. J. Daemen and V. Rijmen. The Rijndael block cipher. AES Proposal, Mar. 1999.
24. I. Damgård, M. Geisler, and M. Krøigaard. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy*, pages 416–430. Springer, 2007.
25. I. Damgård and M. Jurik. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography*, pages 119–136. Springer, 2001.
26. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *Information Theory, IEEE Transactions on*, IT-31(4):469–472, 1985.
27. Z. Erkin. *Secure signal processing: Privacy preserving cryptographic protocols for multimedia*. PhD thesis, Delft University of Technology, The Netherlands, 2010.
28. Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, pages 235–253. Springer, 2009.
29. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):647, 1985.
30. P. Failla. Heuristic Search in Encrypted Graphs. *Emerging Security Information,Systems and Technologies – SECURWARE'10, Fourth International Conference on*, pages 82–87, 2010.
31. P. Failla and M. Barni. Gram - Schmidt Orthogonalization on Encrypted Vectors. In *Digital Communications – ITWDC 2010, 21st International Tyrrhenian Workshop on*, 2010.
32. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Theory of Computing, Proceedings of the 41st annual ACM symposium on*, pages 169–178. ACM, 2009.
33. C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. *Advances in Cryptology–EUROCRYPT 2010*, pages 506–522, 2010.
34. D. Giry and J.J. Quisquater. Cryptographic key length recommendation, 2010.
35. O. Goldreich. *Foundations of cryptography*. Cambridge University Press, 2004.
36. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Theory of computing, Proceedings of the nineteenth annual ACM symposium on*, pages 218–229. ACM, 1987.
37. S. Goldwasser and S. Micali. Probabilistic encryption* 1. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
38. G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins Univ Pr, 1996.
39. P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE transactions on*, 4(2):100–107, 1968.
40. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *LNCS*. Springer, 2003.
41. M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran. On compressing encrypted data. *Signal Processing, IEEE Transactions on*, 52(10):2992–3006, 2004.

42. S. Katzenbeisser, A. Lemma, M.U. Celik, M. van der Veen, and M. Maas. A buyer–seller watermarking protocol based on secure embedding. *Information Forensics and Security – TIFS, IEEE Transactions on*, 3(4):783–786, 2008.
43. N. Koblitz. *A course in number theory and cryptography*. Springer, 1994.
44. V. Kolesnikov, A.R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. *Cryptology and Network Security*, pages 1–20, 2009.
45. V. Kolesnikov, A.R. Sadeghi, and T. Schneider. Modular design of efficient secure function evaluation protocols. Technical report, Cryptology ePrint Archive, Report 2010/079, 2010, http://eprint. iacr. org/2010/079/.[Online]. Available: http://thomaschneider. de/papers/KSS10. pdf, 2010.
46. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming (ICALP'08)*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.
47. Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206, 2008.
48. Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of cryptology*, 22(2):161–188, 2009.
49. Y. Lindell, B. Pinkas, and N. Smart. Implementing two-party computation efficiently with security against malicious adversaries. In *Security and Cryptography for Networks – SCN'08*, volume 5229 of *LNCS*, pages 2–20. Springer, 2008.
50. Y.D. Ma, C.L. Qi, Z.B. Qian, F. Shi, and Z.F. Zhang. A novel image compression coding algorithm based on pulse-coupled neural network and Gram-Schmidt orthogonal base. *Dianzi Xuebao(Acta Electronica Sinica)*, 34(7):1255–1259, 2006.
51. D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. In *USENIX*, 2004. http://fairplayproject.net.
52. C.A. Melchor, P. Gaborit, and J. Herranz. Additively Homomorphic Encryption with t-Operand Multiplications. *Crypto 2010*, 1996.
53. A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC, 1997.
54. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Discrete Algorithms – SODA'01, ACM-SIAM Symposium On*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.
55. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. Cryptology ePrint Archive, Report 2011/091, 2011. http://eprint.iacr.org/.
56. S.J. Orfanidis. Gram-Schmidt neural nets. *Neural Computation*, 2(1):116–126, 1990.
57. M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. SCiFI-a system for secure face identification. In *Security and Privacy, 2010 IEEE Symposium on*, pages 239–254. IEEE, 2010.
58. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT99*, pages 223–238. Springer, 1999.
59. A. Paus, A.-R. Sadeghi, and T. Schneider. Practical secure evaluation of semi-private functions. In *Applied Cryptography and Network Security – ACNS'09*, volume 5536 of *LNCS*, pages 89–106. Springer, 2009. http://www.trust.rub.de/FairplaySPF.
60. B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure two-party computation is practical. In *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *LNCS*. Springer, Dec. 2009. Full version available at http://eprint.iacr.org/2009/314.
61. S. Rane and W. Sun. Privacy preserving string comparisons based on Levenshtein distance. In *Information Forensics and Security – WIFS 2010, IEEE International Workshop on*, pages 1–6. IEEE, 2010.
62. D.K. Rappe. Homomorphic cryptosystems and their applications. *Volume Ph.D.*, 2004.
63. R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–178, 1978.
64. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):126, 1978.

65. A.R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Information Security and Cryptology – ICISC 2009, International Conference on*. Springer, 2009.
66. A. Sharma and K.K. Paliwal. Fast principal component analysis using fixed-point algorithm. *Pattern Recognition Letters*, 28(10):1151–1155, 2007.
67. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43, 2010.
68. A. C. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS'86)*, pages 162–167. IEEE, 1986.
69. A.C. Yao. Protocols for secure computations. In *Foundations of Computer Science, Proceedings of the 23rd Annual IEEE Symposium on*, volume 23, pages 160–164. Citeseer, 1982.
70. W. Zheng, C. Zou, and L. Zhao. Real-time face recognition using Gram-Schmidt orthogonalization for LDA. *Pattern Recognition*, 2:403–406, 2004.