SHE based Non Interactive Privacy Preserving Biometric Authentication Protocols

Giulia Droandi, Riccardo Lazzeretti
Department of Information Engineering and Mathematics
University of Siena, Siena, Italy
giulia.droandi@gmail.com, lazzeretti@diism.unisi.it

***Abstract*—Being unique and immutable for each person, biometric signals are widely used in access control systems. While biometric recognition appeases concerns about password's theft or loss, at the same time it raises concerns about individual privacy. Central servers store several enrolled biometrics, hence security against theft must be provided during biometric transmission and against those who have access to the database. If a server's database is compromised, other systems using the same biometric templates could also be compromised as well. One solution is to encrypt the stored templates. Nonetheless, when using traditional cryptosystem, data must be decrypted before executing the protocol, leaving the database vulnerable. To overcame this problem and protect both the server and the client, biometrics should be processed while encrypted. This is possible by using secure two-party computation protocols, mainly based on Garbled Circuits (GC) and additive Homomorphic Encryption (HE). Both GC and HE based solutions are efficient yet interactive, meaning that the client takes part in the computation. Instead in this paper we propose a non-interactive protocol for privacy preserving biometric authentication based on a Somewhat Homomorphic Encryption (SHE) scheme, modified to handle integer values, and also suggest a blinding method to protect the system from spoofing attacks. Although our solution is not as efficient as the ones based on GC or HE, the protocol needs no interaction, moving the computation entirely on the server side and leaving only inputs encryption and outputs decryption to the client.**

I. Introduction

Biometric signals such as faces, fingerprints and irises are often used in access control systems to authorize users' membership. This avoids problems linked to password leakage and theft but at the same time raises concerns about individual's privacy. Biometric templates are unique for each person; if a control system is compromised, then all the other systems using the same biometrics are compromised as well. Hence in a distributed biometric system with a high level of privacy compliance, privacy protection between the server and the client must also be guaranteed to avoid tracking users' routine. Usually, protection against eavesdropping is guaranteed by traditional encryptions schemes. Regardless of this, several attacks on biometric authentication protocols can be carried out. Among all these possible threats the most studied are *spoofing* attacks [18]. This is where the intruder uses some kind of fake biometric to imitate a legitimate user (creating a fake digital template) to fraudulently access the database or the server. Recent studies have hypothesized the use of biometric samples (taken from social media or stolen from latent prints) to forge fake templates. In fact a biometric template can be encoded in an integer vector of fixed length [1], [11], [12] and usually a distance among the two template vectors is evaluated to verify that they belong to the same individual. Unfortunately, an attacker that has forged a biometric template could use the resulting distance to improve the fake features by performing a hill-climbing attack where the fake template is corrected according the output of previous rounds. For this reason it is important to also keep the resulting distance secret.

The most used approach to protect the privacy of both client and server templates is to implement the matching process by using Secure Two-Party Computation (STPC) protocols [21]. Many biometric protocols [7, chapter 7] have been proposed in the past, based on fingerprints [1], [2], irises [17], face recognition [12], [22], etc. mainly relying on Garbled Circuits theory (GC) [25] and Homomorphic Encryption (HE) [13]. GC provides a generic implementation able to evaluate any function represented through a boolean circuit wherein one party "encrypts" the boolean circuit and another party obliviously evaluates the circuit without accessing the plain intermediate values. HE allows only upon specific operations on ciphertexts, such as the addition in the well known Paillier cryptosystem [19], while others operations can be evaluated by using interactive protocols. GC and HE have also been used in hybrid protocols where HE or GC are chosen according their characteristics to evaluate different parts of the protocol corresponding to different functionalities [3]. Despite the protocols being quite efficient, interaction between the parties involved is needed during protocol computation.

Full Homomorphic Encryption (FHE), allowing both adition and multiplication of encrypted data has been proposed by Gentry [14]. A Somewhat Homomorphic Encryption (SHE) scheme, performing a limited number of additions and multiplications on ciphertexts, is used to subsequently build a FHE scheme, through undertaking a bootstrapping operation. According to Gentry's approach, FHE encrypts single bits, allowing computation of sums and products (modulus 2) between them. Being possible to evaluate XOR and AND binary gates, the universal NAND gate (and hence any boolean circuit) can be evaluated. Even if FHE is far from being as efficient as GC and HE protocols (despite the many optimizations of Gentry's cryptosystem proposed [15], [24]), FHE (and also SHE for light protocols) have raised the interest of researchers because all the computation is moved to the server, leaving only input encryption and output decryption to the client.

In this paper, we propose a new SHE implementation of biometric authentication protocol . The client first enrolls an encrypted feature vector, derived from his biometric template, in the server database. Then, during authentication, the client sends a probe, a new instance of the same biometric, and the

server has to decide if the probe is sufficiently similar (meaning if the distance between the two biometrics is under a certain threshold) to the one stored in database. Usually, Hamming distance is used for those biometrics encoded with bit's vector and Euclidean distance otherwise. The reason for resorting to SHE instead of FHE, is that a solution based on SHE is more feasible, due to the limited number of operations performed on the encrypted data, and hence also more efficient. Moreover, to reduce the number of products we use a SHE implementation that can encrypt an integer number in a given interval. More specifically, our protocol relies on the application of the SHE scheme proposed by Pisa et al. in [20] into the biometrics recognition problem. As a further contribution, we test the general protocol on iris [11], [17] and fingerprint recognition [1], [2] (but it can work on other biometrics as well). We also propose a blinding method to protect computation's distance output from spoofing attacks. Further, as a side contribution, we propose a way to extend the use of the scheme proposed in [20] to negative numbers and use it to reduce the number of multiplication in distances computation. To the best of our knowledge, before this paper, a SHE solution has been applied to biometric recognition only in [23] and in [26]. In respect to them, we apply our protocol to two different templates, iris and fingerprints, and we are able to confront the resulting distance with threshold in encrypted domain without using a third party or interactive protocols. Moreover we blind server outputs distance in order to prevent spoofing attacks.

The paper is organized as follows. In Section II, we describe the extension of Pisa et al. [20] that makes it possible to operate on more than one bit at a time. In Section III, we outline the working principles of the biometric-based authentication systems and we describe a new protocol putting in practice the theoretical principles underlying the proposed extension. In Section IV, we validate experimentally the protocol. Finally, in Section V we draw some conclusions and we present some possible future works.

II. Cryptographic Primitives

An encryption scheme is defined *Fully Homomorphic* if it is capable of performing implicit plain-text addition and multiplication by manipulating only the ciphertexts. The first construction of a Fully Homomorphic scheme has been provided by Gentry [14] who proposed a scheme based on the use of lattices and additional noise.

A FHE scheme is the result of three steps. The first step consists in constructing a *Somewhat Homomorphic Encryption* (SHE) scheme. In this sort of encryption schemes, the message is usually hidden with some *noise*. SHE scheme can perform only a limited number of operations before incurring decryption errors due to the noise that increases following each operation, especially multiplications. The second step is the *squash*. This consists in representing the decryption function as a low degree polynomial that can be applied to encrypted data and secret key bits. When the noise becomes too large, the decryption performed during the squashing stage becomes impossible. Gentry's intuition was to tackle with this problem by introducing an additional third step, called *bootstrapping*. This consists in the homomorphic evaluation of the polynomial by implementing the decryption on ciphertexts and additional encrypted key bits. By doing so, a new (refreshed) ciphertext of the same bitlength is produced, but with reduced noise. Intuitively, during bootstrapping the ciphertext is re-encrypted with less noise. As a consequence, the ciphertext with reduced noise can undergo further operations before again incurring into errors. By repeating the bootstrapping step before the noise becomes too large, the number of possible operations becomes virtually unlimited, thus allowing the construction of a fully homomorphic scheme. In subsequent years, a number of variants improving Gentry's original idea have been devised, including: algorithmic optimizations [15]; Van Dijk et al. variant implemented on integers ring [24], followed by its improvement with shorter public keys [9], [10] and batch version [8]; the schemes based on Learning With Error (LWE) [5] and Ring Learning With Error (RLWE) [6]; plus many others.

A. Extension of DGHV to the Integers

For our application, a FHE scheme is not really necessary, because beforehand we know the number of operations that will be performed. Hence, we focus on the SHE scheme described in [20] that does not need the expensive squash and bootstrapping operations. In this section, we present the extension of Van Dijk et al. scheme (DGHV scheme [24]), applied to integer numbers due to Pisa et al. in [20].

Given two integer numbers x and p, we indicate with $[x]_p$ and $(x \mod p)$ the reductions of x modulo p, where $[x]_p$ denotes the remainder r in the interval $[0, p)$, while $x \mod p$, refers to the integer in the interval $(-p/2, p/2]$, i.e. $x \mod p = [x]_p - p$ when $[x]_p > p/2$.

In the original DGHV scheme, only binary messages can be encrypted. Given a message m, it is encrypted as $c_{\text{dghv}} = m + pq + 2r$ for some $q, r \in \mathbb{Z}$, where r is the added *noise*. Starting from here, Pisa et al. adapt the encryption function as $c = m + p \cdot q + br$, with $m \in [0, b)$, where the integer p must be not divisible by the base b. Even if b could be any integer we will consider only bases that are powers of 2, i.e. given a bitlength k, then $b = 2^k$. Our small contribution to the cryptographic scheme is the extension to negative numbers. Given the base b, we can encrypt integers in the interval $[-b/2, b/2)$. In this case the decryption function is performed as $([c]_p \mod b)$. Obviously to represent negative numbers the base b should be twice the maximum integer that can be obtained during the computation.

We now define the scheme presented in [20] by Pisa et al.:

`KeyGen`(λ). The secret Key is an integer $p \in [b^{\eta-1}, b^{\eta}) \cap \mathbb{Z}$ such that p is not divisible by b. The public key is a set of τ elements obtained as $x_i = q_i \cdot p + r_i$ for all $0 \leq i \leq \tau$, where $r_i \in (-b^{\rho}, b^{\rho})$ is a randomly chosen noise and $q_i \in [0, b^{\gamma}/p)$. The element x_0 must be greater than every other x_i, it must be odd and the noise must be even. The public key is $\mathbf{pk} = \{x_0, x_1, \ldots, x_{\tau}\}$.

`Encrypt`$(\mathbf{pk}, m)$. Given an integer message $m \in [0, b)$ (or $[-b/2, b/2)$ in our solution coping with negative numbers), choose a random integer $r \in (-b^{\rho'}, b^{\rho'})$ and let S be a sparse subset of indexes in $\{1, \ldots, \tau\}$. The ciphertext c is:

$$c = \left[m + br + b \sum_{i \in S} x_i \right]_{x_0} . \tag{1}$$

λ	Base	Secret Key	Cipher	Public Key
10	2	10 B	0.1 KB	0.1 MB
	2^{50}	506 B	6.3 KB	6.6 MB
	2^{100}	1013 B	12.5 KB	13.8 MB
	2^{150}	1519 B	18.8 KB	21.7 MB
15	2	25Byte	7 KB	7 MB
	2^{50}	1 KB	360 KB	352 MB
	2^{100}	2.4 KB	721 KB	704 MB
	2^{150}	3.5 KB	1081 KB	1056 MB
20	2	45 Byte	66 KB	216 MB
	2^{50}	2.2 KB	3 MB	11 GB
	2^{100}	4.5 KB	6 MB	21 GB
	2^{150}	6.7 KB	10 MB	32 GB

TABLE I. SIZE OF THE CIPTHERTEXT, SECRET AND PUBLIC KEY IN FUNCTION OF THE SECURITY PARAMETER AND THE BASE.

`Decrypt`(p, c). Given a ciphertext c, the decryption function is: $m = \left[[c]_p\right]_b$ if only positive numbers are needed, $m = [c]_p \mod b$ in our extension to negative numbers.

`Evaluate`$(\mathbf{pk}, C, c_1, \ldots, c_t)$. By using the SHE scheme, any circuit C composed by algebraic gates (additions and products) can be evaluated. The addition and multiplication between two messages $m_i,\ m_j$ are mapped in the addition and multiplication between the relative ciphertexts c_i, c_j, modulus x_0.

B. Parameters

As in [24] and [20], we define the following parameters: the security parameter λ; the bitlength η of the secret key; the bitlengths ρ and ρ' of the noise in the public key and in the ciphertext encryption respectively; the number τ of integers of the public key. .

In order to preserve the semantic security of the reduction to approximate-GCD problem as in [24], the parameters depend on λ polynomially. According to [20]: $\rho = \lambda$, $\rho' = 2\lambda$ respectively for the noise of each element of public key and ciphertext; $\eta = O(\lambda^2)$ for the length of the private key; $\gamma = O(\lambda^5)$ to define the length of the ciphertext; $\tau = \gamma + \lambda + \log_2(b)$ is the number of elements of the public key.

The SHE scheme described above allows only a limited number of operations on encrypted data. In order to decrypt the message correctly, the total noise should not grow more than $p/2$. While after a multiplication we have a significant noise increase, addition is less problematic because it produces only a slight increase. Determining the maximum number of possible multiplications means finding the largest μ that satisfies $R^\mu < \frac{p}{2}$, where $R = b^{2\lambda+1} + \lambda^5 b^\lambda (b^2 + b)$ is the maximum allowed noise, yielding $(b^{2\lambda+1} + \lambda^5 b^\lambda (b^2 + b))^{\mu+1} < \frac{p}{2}$. Similarly we can obtain the limit for the maximum number of additions allowed by the scheme. As shown in [20], the number of multiplications depends mainly on the security parameter λ while they are quite independent from the base. Only few multiplications can be evaluated before a decryption error occurs, while the maximum number of additions is pretty high. For the security basis of the scheme we refer to [20]. As it is possible to see from Table I, public key is very expensive in terms of memory. This had led to problems during the implementation. Indeed for security we need large λ, but this implies big public keys and therefore memory problems.

III. BIOMETRIC RECOGNITION

We here describe the general privacy preserving biometric authentication protocol, based on SHE cryptosystem working on integers. In an authentication protocol there are two parties involved, client $\mathcal{C}$ and server $\mathcal{S}$, and the client has to prove that he is who he claims to be. First, during an enrollment phase, the client generates a feature vector from his biometric and sends it encrypted with his public key to the server for future use as template. To him, the server gives back an identification tag. This could be the client's username or a progressive number. Along with the template, the client sends his public key (the server would need it to perform operations on features).

We consider each biometric template represented as vector of n elements belonging to a fixed integer interval (usually $[0, 1]$ for iris and $[0, 2^\ell]$ for some $\ell \in \mathbb{N}$ for fingerprints. See Section IV for details). The authentication protocol is evaluated in three steps (Figure 1).

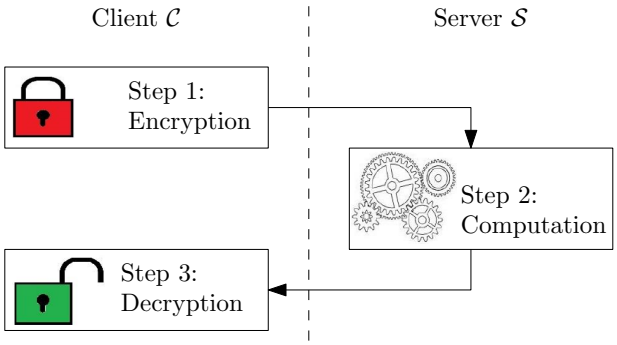

Fig. 1. The three steps of the protocol.

Step one: Client encryption. The client generates a probe $\mathbf{q} = (q_1 \ldots q_n)$ from his biometric and sends it encrypted to the server along with his identification tag.

Step two: Server computation. The server $\mathcal{S}$ runs a privacy preserving protocol to calculate the Hamming or Euclidean distance, $D(\mathbf{q}, \mathbf{s})$, between the probe and the features stored in the server database, $\mathbf{s} = (s_1, \ldots, s_n)$(Sections III-1,III-2). Then, given a similarity threshold ϵ, it must verify that $D(\mathbf{q}, \mathbf{s}) < \epsilon$. To evaluate the comparison, we should pass from an integer representation of the value to a binary representation. Unfortunately this is not possible without interacting with the secret key owner. To bypass the problem the server computes $d = D(\mathbf{q}, \mathbf{s}) - \epsilon$ and opportunely blinds the obtained integer d (Section III-3) without changing its sign and sends it back to client. In fact the result of the authentication protocol corresponds to the sign of d, but the magnitude needs to be obfuscated to protect the database from malicious attacks (i.e. spoofing).

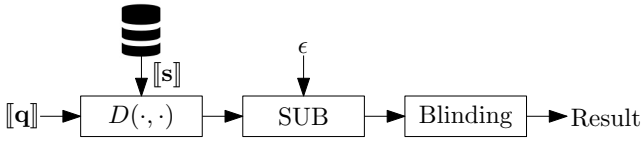

Fig. 2. Step 2:Server computation.

Step three: Client decryption. The client decrypts the received number. If it is a negative number he has demonstrated that he is who he claims.

In the following, we investigate the SHE-based implementation of protocols necessary for biometric authentication: Hamming Distance, Euclidean Distance, Threshold and Blinding. Given a message m we will indicate with $[\![m]\!]$ its encryption with client public key. If $\mathbf{t}$ is a n elements vector of messages then $[\![\mathbf{t}]\!]$ is the element-wise encryption $([\![t_1]\!], \ldots, [\![t_n]\!])$ of $\mathbf{t}$. We remind that the representation of negative numbers is possible as described in Section II.

1) Hamming Distance: If the features are binary values, the Hamming distance $HD(\mathbf{q}, \mathbf{s}) = \|(\mathbf{q} \otimes \mathbf{s})\|$, is used ($\otimes$ identifies the XOR and $\|\cdot\|$ the norm of the binary vector). The XOR could be easily implemented through an addition in a base 2 SHE scheme, but in our implementation a greater base is used to facilitate the following sum computation. We hence need to operate some necessary changes to fit the distance's calculus to our encryption scheme. Since we are working with integers and our scheme can perform additions modulus b, the Hamming distance can be calculated as $HD(\mathbf{q}, \mathbf{s}) = \sum_{i=1}^{n} s_i + q_i - 2q_i s_i = s_i + q_i(1 - 2s_i)$. Every term of the previous formula is encrypted with user's public key, even $1 - 2s_i$. This last term could be either sent to server in enrollment phase or calculated in advance by the server as $[\![1]\!] + [\![-1]\!] \cdot [\![s_i]\!]$. In practice, after $\mathcal{S}$ have received the encryption of the probe $[\![\mathbf{q}]\!]$ the server computes

$$\begin{aligned} [\![HD(q,s)]\!] &= [\![\sum_{i=1}^{n} q_i \cdot (1 - 2s_i) + s_i]\!] \\ &= \sum_{i=1}^{n} \left([\![q_i]\!] \cdot [\![1 - 2s_i]\!] + [\![s_i]\!]\right). \end{aligned} \quad (2)$$

2) Euclidean Distance: If the features are integers values the squared Euclidean distance $ED(\mathbf{q}, \mathbf{s}) = \left\|(\mathbf{q} - \mathbf{s})^2\right\|$ is used. To compute the lowest possible number of multiplication, the server stores in its database $[\![-\mathbf{s}]\!]$. It could either have been sent in the enrollment phase or it could have been calculated as $[\![-1]\!] \cdot [\![s_i]\!]$ from the encryption of client's template. Therefore given the client probe, the server actually performs:

$$[\![ED(q,s)]\!] = \sum_{i=1}^{n} \left([\![q_i]\!] + [\![-s_i]\!]\right)^2. \quad (3)$$

3) Thresholding and blinding : Now the distance $D(\mathbf{q}, \mathbf{s})$ is compared with the threshold ϵ, by computing the difference between the distance and the threshold under encryption, i.e. $[\![d]\!] = [\![D(\mathbf{q}, \mathbf{s}) - \epsilon]\!] = [\![D(\mathbf{q}, \mathbf{s})]\!] + [\![-\epsilon]\!]$. The sign of d corresponds to the result of the authentication protocol.

In order to protect the biometrics enrolled in the database, the number $[\![d]\!]$ has to be blinded, before being disclosed to the client. Unfortunately, additive blinding cannot be used because it could change the sign of $D(\mathbf{q}, \mathbf{s}) - \epsilon$. On the other hand, multiplicative blinding is known to be much less secure [4]. The best solution, among those affordable by the encryption scheme in use, is to adopt a hybrid multiplicative/additive blinding: in practice two random values k_1 and k_2 are chosen so that the final result is obfuscated by the server through computing $[\![k_1(D(\mathbf{q}, \mathbf{s}) - \epsilon) + k_2]\!] = [\![k_1]\!][\![D(\mathbf{q}, \mathbf{s}) - \epsilon]\!] + [\![k_2]\!]$ and the sign of the result is not changed.

Given the base b, the range of possible values is $[-b/2, b/2)$, hence k_1 has to be chosen so that the product modulus doesn't exceed $b/2$, limiting the possible values to the range $\left[1, \left\lfloor \frac{b}{2(D_{\max} - \epsilon)} \right\rfloor\right)$, where $D_{\max}$ denotes the maximum value the distance can assume. Then also k_2 must be chosen in such a way that the sign of the final result does not change. First of all k_2 can be positive or negative. To define its range we have to address the two worst cases we can encounter: i) if $D(\mathbf{q}, \mathbf{s}) - \epsilon = 1$, then $|k_2| < k_1$; ii) if $D(\mathbf{q}, \mathbf{s}) = D_{\max}$, then $|k_2| < b/2 - k_1(D_{\max} - \epsilon)$. By defining $k_{2,\lim} = \min\{k_1, b/2 - k_1(D_{\max} - \epsilon)\}$, k_2 is randomly chosen in the range $(-k_{2,\lim}, k_{2,\lim})$. To guarantee high security levels, a very large base would be needed, however this would result in a too complex system so a trade-off is needed (Section IV for more details about the trade-off we reached in our system).

A. Improvements with respect to classical binary SHE implementation

A biometric authentication protocol can also be easily implemented by relying on the classic binary DGHV scheme. In the Hamming Distance, the XORs among the bits could be implemented without products, but the sum of the n $q_i \oplus s_i$ elements requires a circuit composed by many AND gates. In fact a reverse tree structure, is composed by $\log_2 n$ layers would be used, wherein the i-th layer is composed by $n/2^i$ addition circuits [16] of i-bit long inputs (each one needing i AND gates[1]). Hence $2(n-1) - \log_2 n$ products are needed for the sum and being the depth of the tree $\log_2 n$, it is important that at least $\log_2 n$ multiplications are allowed before the SHE incurs in a decryption error. A greater number of products would be needed in a binary based Euclidean distance.

On the other side, since our solution works directly on integer values, only n products are evaluated in parallel to compute the XORs ($q_i \cdot (1 - 2s_i)$), plus one product for the blinding. Also in Euclidean distance, n products are sufficient to evaluate in parallel the square of $q_i - s_i$ and one for the blinding. Hence in the computation of both the distances it is sufficient that the SHE scheme allows the evaluation of at least 2 products, if we consider that $[\![1 - s_i]\!]$ or $[\![-s_i]\!]$ have been sent encrypted from client in enrollment phase, while if the server has to compute them before executing the protocol, SHE must deal with three multiplications.

IV. Technical results

The SHE-based iris and fingerprint authentication protocols have been implemented and tested on a desktop equipped with a Quad-Core CPU (Intel i7 at 3,40GHz) and 16 GB RAM, mounting a 64-bit Windows OS, to measure the communication and computational complexity of the scheme in terms of bandwidth and protocol runtime. To optimize the protocol, distance computations have been parallelized into 4 threads, as allowed by our system characteristics.

[1]The addition circuit in [16] is designed for a GC implementation but can be evaluated also in SHE protocols.

λ	Base	Public Key	Encryption	Decryption	Multiplication
10	2	0.01 s	0.20 ms	0.00 ms	0.00 ms
	2^{50}	0.58 s	0.27 ms	0.93 ms	3.00 ms
	2^{100}	2.03 s	0.63 ms	2.57 ms	8.07 ms
	2^{150}	3.59 s	0.93 ms	4.80 ms	15.13 ms
15	2	0.18 s	0.60 ms	0.00 s	0.01 s
	2^{50}	2 min 39 s	15.68 ms	0.18 s	1.11 s
	2^{100}	4 min 42 s	28.58 ms	0.45 s	3.01 s
	2^{150}	8 min 56 s	44.00 ms	0.75 s	5.66 s

TABLE II. TIMES NEEDED TO INITIALIZE THE SYSTEM

	λ	Base		
		2^{50}	2^{100}	2^{150}
Iris	10	13 MB	25 MB	38 MB
	15	750 MB	1.4GB	2 GB
Finger	10	606 KB	1.2 MB	1.7 MB
	15	34 MB	68 MB	102 MB

TABLE III. IRISCODE AND FINGERCODE TEMPLATES COMMUNICATION COMPLEXITY.

λ	Base	Step 1 Client Encryption	Step 2 Server Computation	Step 3 Client Decryption
10	2^{50}	1.2 s	2.0 s	0.2 ms
	2^{100}	1.2 s	5.3 s	4.3 ms
	2^{150}	1.8 s	9.7 s	5.6 ms
15	2^{50}	29s	14 min 33 s	0.2 s

TABLE IV. IRISCODE PROTOCOL'S AVERAGE EXECUTION TIME

λ	Base	Step 1 Client Encryption	Step 2 Server Computation	Step 3 Client Decryption
10	2^{50}	0.03 s	0.10 s	1 ms
	2^{100}	0.06 s	0.30 s	3 ms
	2^{150}	0.09 s	0.53 s	5 ms
15	2^{50}	1.37 s	37 s	0.18 s
	2^{100}	2.80 s	1min 48 s	0.45 s
	2^{150}	4.25 s	3 min 17 s	0.79 s

TABLE V. FINGERPRINT PROTOCOL'S AVERAGE EXECUTION TIME

According to Sections II-A and II-B, the secret key is in the range $[b^{\eta-1}, b^{\eta})$, with $b = 2^k$ and $\eta = \lambda^2$, and hence is represented with $k\eta$ bits. Similarly the secret key and each element composing the public key are represented with up to $k\lambda^5$ bits. Table I shows how the sizes of ciphertext, public key and secret key increase as a function of the base b and the security parameter λ. To guarantee a sufficient security level in biometric recognition, high values of λ should be used. However, for blinding we need at least $b = 2^{50}$, for which, if $\lambda = 20$, we obtain a public key of 11GB, hence for our tests we considered $\lambda = 10, 15$ for which public key is about 7MB and 350MB respectively. We ran 150 tests and we have measured the average times (ms or s) required by single encryption, decryption, key generation and multiplication, by using a Java implementation of the SHE scheme. The averaged results of the cryptosystem are shown in Table II. Addition and secret key generation are not reported, because their runtimes are negligible. The runtime of all the other operations grows with the base's bit-length and above all the security parameter. For high values of b and λ, public key generation is an expensive operation, but it must be executed only one time by the client and it does not affect the authentication protocol.

Iris protocol. We considered the algorithm proposed in [11], [17], where an iris is represented through a vector of 2048 binary features, hence the Hamming distance is used and calculated as described in Section III-1. For $\lambda = 10$ we have run the protocol on different bases $2^{50}, 2^{100}, 2^{150}$. To perform blinding, for $b = 2^{50}$ the choice of k_1 is in the range $\left(0, 2^{39}\right)$, for $b = 2^{100}$ $k_1 \in \left(0, 2^{88}\right)$, while for $b = 2^{150}$ $k_1 \in \left(0, 2^{110}\right)$. As shown in Table III with $\lambda = 15$, memory occupied by a single iris template grows significantly with base's increase (for a 150 bits base it takes about 2GB) and during server computation three 2048-elements vectors must be memorized at the same time (that needs about 6 GB) plus the public key (about 2GB). Due to those memory problems, we have run test only for $b = 2^{50}$. 2048 ciphertexts are transmitted from the client to the server and a single ciphertext from the server to the client, resulting in the total transmission of 13MB, 25 MB and 38MB respectively for each base with the lowest security parameter considered, while with the biggest λ considered the total transmission is 750MB (Table III). We repeated the tests on iris protocols 50 times and the average results for each part of the protocol are shown in Table IV. As expected, independently from the parameter set, the most expensive part is server's one. With $\lambda = 10$ the whole computation runtime takes some seconds (from about 3 to 17 seconds), while with $\lambda = 15$ server computation is really slow, it takes about 14 minutes (we remember that only $b = 2^{50}$ has been tested for memory problems).

Fingerprint protocol. We considered the system described in [1], [2], where the authors demonstrate that the representation of a fingercode through a vector of 192 features of 4 bits guarantees an equal error rate of 6.7%. The implementation requires the computation of the squared Euclidean distance, whose maximum value is 43200, hence 16 bits are needed for its representation. We also observe that according to the results reported in [1], [2], by representing a fingercode with a vector of 96 features of 2 bits each, the equal error rate increases to only 7.6%, while the maximum distance value decreases to 864. This can be represented with 10 bits, allowing a larger range[2] for k_1 and making the configuration more appealing for a privacy preserving SHE implementation. For each security parameter's value we run the tests using different bases $2^{50}, 2^{100}, 2^{150}$. The vector length of fingercode is less than iriscode, allowing us to use a bigger base in tests (for $\lambda = 15$ and $b = 2^{150}$ public key is about 1GB, a iriscode feature about 2GB while a fingercode feature is only about 101MB). In this case 96 ciphertexts are transmitted from client to server and one from server to client, with a bandwidth of about 102 MB in the biggest parameter set considered. Bandwidth for different setups of bases and λ's is summarized in Table III. As for the iriscode case, the most expensive part is server's computation, which takes up to 3 minutes. In this case execution time is faster than in the iris case, due to the lower number of element's for each fingerprint representation. Results obtained for fingerprint matching are summarized in Table V.

As expected the runtimes needed by the SHE implementation of the privacy preserving iris and fingerprint matching protocols are by far larger than the execution time of protocols based on Paillier HE or GC. Nonetheless in our protocol all the computation is moved onto the server side and no interaction is needed. Moreover, runtimes can be lowered by using powerful

[2]The range of k_1 in each base set is similar to the Iris case.

servers allowing for parallelization across more threads. We also underline that respect to the C++ implementation of the face recognition protocol proposed in [23], our implementation is faster or at least comparable, especially for fingerprints and for some parameter sets on the iris case, (their implementation runs in 12.3 seconds). Respect to [23], [26], we also propose a solution (even if not perfectly secure) that obfuscates the final result, while in [23] the final result is disclosed to the client or to a third party, making their solution weak against spoofing attacks.

V. Conclusion

We have implemented and tested a recently proposed SHE scheme [20] that works on integers values. Then we used it to build a privacy preserving biometric matching protocol. The resulting complexity is lower than that of a SHE solution working only on bits. Having prior knowledge of the number of multiplication required by the protocol allows us to use a SHE scheme instead of a FHE, avoiding in this way expensive squash and bootstrap operations. Even if the protocol is not as efficient as protocols based on other STPC techniques such as Garbled Circuits and Homomorphic Encryption our solution has the advantage of moving all computation to the server side, without the necessity of interaction together with the biometric source, hence making it suitable for computation with low power devices. We have observed, from tests, that operation runtime and bitsize of ciphertext and keys are more affected from the security parameter than the base. Concerning the iris and fingerprint authorization protocols, we have tested them with different security parameters and observed that matches take from some seconds to several minutes, especially in the iris case.

In the future, we are interested to improve the protocol by using innovative solutions that everyday are proposed on FHE schemes. That is, to verify if a change of the base during the protocol would make it possible to switch from base b to base 2 and implement a comparison with the threshold through a binary circuit composed by AND and XOR gates.

References

[1] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida 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.

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

[3] Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, A Sadeghi, and Thomas Schneider. Privacy-preserving ecg classification with branching programs and neural networks. *Information Forensics and Security, IEEE Transactions on*, 6(2):452–468, 2011.

[4] T. Bianchi, A. Piva, and M. Barni. Analysis of the security of linear blinding techniques from an information theoretical point of view. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5852–5855, May 2011.

[5] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 97–106. IEEE, 2011.

[6] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology–CRYPTO 2011*, pages 505–524. Springer, 2011.

[7] P. Campisi. *Security and Privacy in Biometrics*. Springer, 2013.

[8] J. Hee Cheon, J.S. Coron, J. Kim, M.S. Lee, T. Lepoint, M. Tibouchi, and A. Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2013*, pages 315–335. Springer, 2013.

[9] J.S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology–CRYPTO 2011*, pages 487–504. Springer, 2011.

[10] J.S. Coron, D. Naccache, and M. Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2012*, pages 446–464. Springer, 2012.

[11] J. Daugman. How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 4:21–30, Jan. 2004.

[12] 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.

[13] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007, 2007.

[14] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178. ACM, 2009.

[15] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. *Advances in Cryptology–EUROCRYPT 2011*, pages 129–148, 2011.

[16] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security*, pages 1–20. Springer, 2009.

[17] Y. Luo, S. S. Cheung, T. Pignata, R. Lazzeretti, and M. Barni. An efficient protocol for private iris-code matching by means of garbled circuits. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2653–2656. IEEE, 2012.

[18] T Matsumoto. Artificial irises: importance of vulnerability analysis. In *Proc. Asian Biometrics Workshop (AWB)*, volume 45, 2004.

[19] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyEUROCRYPT99*, pages 223–238. Springer, 1999.

[20] Pedro Silveira Pisa, Michel Abdalla, and Otto Carlos Muniz Bandeira Duarte. Somewhat homomorphic encryption scheme for arithmetic operations on large integers. In *Global Information Infrastructure and Networking Symposium (GIIS), 2012*, pages 1–8. IEEE, 2012.

[21] M. M. Prabhakaran and A. Sahai. *Secure Multi-party Computation*. IOS Press, 2013.

[22] A.R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Information, Security and Cryptology–ICISC 2009*, pages 229–244. Springer, 2010.

[23] J.R. Troncoso-Pastoriza, D. Gonzalez-Jimenez, and F. Perez-Gonzalez. Fully private noninteractive face verification. *Information Forensics and Security, IEEE Transactions on*, 8(7):1101–1114, July 2013.

[24] 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.

[25] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of computer science*, 1982.

[26] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Packed homomorphic encryption based on ideal lattices and its application to biometrics. In *Security Engineering and Intelligence Informatics*, pages 55–74. Springer, 2013.