Non-Interactive Privacy Preserving Protocol for Biometric Recognition Based on Somewhat Homomorphic Encryption

Droandi Giulia
Department of Information Engineering and Mathematics, University of Siena, Siena, Italy
Giulia.droandi@gmail.com

Abstract.
Biometric signals are often used in access control systems because of their immutable and highly discriminative characteristics. While the deployment of biometric access control systems allows user identification without the risk of password leakage or theft, at the same time it raises serious concerns about the leakage of individuals' privacy.
A number of privacy preserving protocols have been proposed to guarantee users' privacy against a centralized database owner. Until now, mainly interactive protocols based on Garbled Circuits (GC) and additively Homomorphic Encryption (HE) have been presented.
In this paper we describe a non-interactive protocol for privacy preserving biometric matching, whose complexity is totally moved to the server side. Only input encryption and output decryption are performed by the client.
This is made possible by relying on a Somewhat Homomorphic Encryption (SHE) scheme, properly modified to handle integer values.
Due to the characteristic of the chosen cryptosystem, it can be applied to many different biometrics, such as iris images and fingerprints. Comparison within vectors is done by using Hamming or Euclidean distance, depending on the biometric used.
Since, in the encrypted domain, multiplication is expensive in terms of computation and time complexity, we have devised a solution that reduces the amount of multiplications. Moreover, several distances are evaluated in parallel to decrease the time complexity of the system. Furthermore, our solution has the advantage of moving all computation to the server side, eliminating the necessity for interaction with the private key owner.

In identification scenarios, client biometric features must be compared with a whole database, owned by the server. This would lead to the necessity of storing the whole database encrypted with a user's public key. We also devise a solution to avoid this necessity.
The new SHE-based protocol proposed has been implemented and tested. Results show that, even if the protocol is not as efficient as the interactive protocols based on GC or additively HE, a non-interactive solution based on SHE is feasible.

Keywords: Biometric, Iriscode, Fingercode, Somewhat Homomorphic Encryption, Privacy Preserving Protocol.

1.Introduction

Because of their unique and immutable characteristics, biometric signals such as faces, iris, and fingerprints, are more and more used by access control systems to authorize users' membership. Biometric access control systems allow user identification without risk of password leakage or theft. Yet it brings with it the necessity of protecting an individual's privacy (Campisi, 2013). Security against eavesdroppers is usually guaranteed by traditional encryption schemes. But 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 user's routine. To reduce concerns over privacy loss, it is necessary to process biometric data in a privacy preserving way. In order that the client does not learn anything about the biometric templates stored in the database, (except for the resulting matching process). In the same way, the biometric server should not be able to extrapolate any information about the query itself or its result.
One approach to protect the privacy of both the client probe and biometric database is to implement the matching process by using Secure Multi-Party Computation (SMPC) protocols (Prabhakaran & Sahai, 2013)(Bringer et al. 2013).

Many previously proposed protocols based on fingerprints (Barni, et al., 2010)(Blanton et al., 2011), irises (Luo, et al., 2012)(Blanton et al., 2011), face recognition (Erkin, et al., 2009)(Sadeghi et al. 2010)(Bringer et al. 2014), etc. mainly rely on Garbled Circuits (GC) (Yao, 1986), Oblivious Transfer (OT) (Rabin, 1981) and additive Homomorphic Encryption (HE) (Fontaine & Galand, 2007).
Unfortunately, GC, OT and HE protocols are interactive; hence they require data exchange between the parties' involved throughout all protocol phases. It is our goal to develop non interactive protocols,

wherein all the computation is performed on the server side. This is possible thanks to Fully Homomorphic (FHE) encryption schemes, capable of performing implicit plain-text addition and multiplication by manipulating only the ciphertexts. This was the most important challenge of modern cryptography. The first construction of a Fully Homomorphic scheme is credited to Gentry (Gentry, 2009) who proposed a scheme based on the use of lattices and additional noise. In subsequent years, a number of variants improving Gentry's original idea were devised. These included: algorithmic optimizations (Gentry & Halevi, 2011); Van Dijk et al. variant working over the integers ring (Van Dijk, et al., 2010) followed by its improvement with shorter public keys (Coron, et al., 2011) and batch version (Cheon, et al., 2013); schemes based on Learning With Error (LWE) (Brakerski & Vaikuntanathan, 2014) and Ring Learning With Error (RLWE) (Brakerski & Vaikuntanathan, 2011); plus many others. All the FHE schemes are based on a Somewhat Homomorphic Scheme (SHE), a cryptographic scheme that undertakes a limited number of multiplications and additions in the encrypted domain. FHE goes beyond this barrier, at the expense of huge communication and computation conplexity.

The implementation shown in this paper is an example of non interactive identification protocol based on SHE. The reason for resorting to SHE instead of FHE, is that due to the limited number of operations performed on encrypted data, a solution based on SHE is not only feasible, but also by far more efficient. In this way all the computation is moved on to the sever side, leaving only input encryption and output decryption to the client. Given that the client does not take part in the computation, the server could also change its tools without providing any information to the client. Our protocol relies on the application of the SHE scheme proposed by Pisa et al. (Pisa, et al., 2012) to the biometrics identification problem. This allows the identification of the presence of a user biometric in a database of templates, without disclosing the plain biometric of the user, and of the database, to each other. As a further contribution, the protocol is applied to iris (Daugman, 2004)(Luo et al.2012) and fingerprint recognition (Jain et al., 2000)(Barni, et al., 2010). Moreover, we propose a way to extend the use of Pisa's et al. scheme to negative numbers.

Concerning the biometrics protocols, we devised a solution allowing computation of Hamming and Euclidean distances with the lowest possible amount of multiplications in the encrypted domain. At the same time, this reduces the overheads of expensive encryptions on the server side without the need of keeping the whole database encrypted with the public key of each user.

Moreover, computation of distances is highly parallelized in such a way to decrease the time complexity of the system. To the best of our knowledge, before this paper, a SHE solution has been applied to biometric recognition in (Troncoso-Pastoriza, et al., 2013) and in (Yasuda, et al., 2013). In both papers the authors use extended or modified versions of the Gentry scheme described in (Gentry & Halevi, 2011). Their aim is to verify if the query submitted is close enough to another encrypted biometry previously enrolled by the user. In the first paper they extended the Gentry scheme to work with positive integer values and apply it to facial recognition. In the second paper they use a variant of Gentry's scheme to pack data and implement a generic biometric recognition protocol.

The paper is organized as follows. In Section 2, we describe the extension of Pisa et al. In Section 3, we outline the working principles of biometric-based authentication systems and we describe a new protocol putting in practice the theoretical principles underlying the proposed extension. In Section 4 we validate the protocol experimentally. Finally, in Section 5, we draw some conclusions and present some possible future works.

2. Primitives.

In this section, we present details of the Somewhat Homomorphic scheme (SHE) that is used to implement our biometric privacy preserving authentication. Pisa's scheme (Pisa, et al., 2012) is an extension of the one presented by Van Dijk et al. usually called DGHV (Van Dijk, et al., 2010). The new scheme works on integer numbers instead of bits. In the following, given two integer numbers x and p, we indicate with $[x]_p$ or $x \bmod (p)$ the reduction of x modulo p. We consider two different modulus operators: the first one $[x]_p$, indicates the remainder r in the interval $[0, p)$, the second one, $x \bmod (p)$, refers to the integer in the interval $(-p/2, p/2]$, i.e. $x \bmod p = [x]_p - p$ when $x \bmod (p) > p/2$.

In the original DGHV scheme (Van Dijk, et al., 2010), only messages of one bit can be encrypted.

Thus the scheme is not suitable for those applications requiring operations on integers. The main improvement, in Pisa's scheme, is that it can encrypt every integer number greater than or equal to zero and less than a certain integer b, called base. Recently, public-key setup has been attacked in (Cohn & Heninge, 2011) so we rely on the more recent protocol described by (Pisa, et al., 2012).
As in the original scheme, Pisa et al. hide the message m in the noise of a near multiple of an integer p i.e. $c = m + pq + br$, where the integer p must not be divisible by b. It is worth noting that the above scheme can be extended and used to encode negative numbers, allowing the encryption of integers in the interval $[-b/2, b/2)$. Obviously, in the case of negative numbers, the base should be twice the maximum integer that needs to be computed.
We now define the scheme presented in (Pisa, et al., 2012).
The secret key of the encryption scheme is an integer p, not divisible by the base and belonging to the interval $[b^{\eta-1}, b^{\eta})$. *The public key* is a set of τ integers obtained as $x_i = r_i + pq_i$, where all r_i and q_i are randomly chosen. The first element, called x_0, must be greater than all the others, odd and its noise must be even.
Given an integer message m in the right interval, the *encryption function* is computed as

$$\left[c = m + br + b\sum_{i=0}^{\tau} x_i\right]_{x_0}$$

To decrypt the ciphertext, it is sufficient to compute two times the modulus operation, the first by the secret key (sk) and the second by the base i.e. $m^* = [c]_{sk} mod(b)$. The scheme is homomorphic, so additions and multiplications on integer values are evaluated by adding or multiplying the corresponding ciphertexts modulus x_0.

Table 1: Parameters.

Cryptosystem parameters	
b	Cryptosystem's base
λ	Security parameter.
η	Bit length of the secret key.
ρ, ρ'	Bit-length of the noise r, in the public key and in the ciphertext encryption.
τ	Number of elements of the public key.

In order to preserve the semantic security of the reduction to approximate-GCD problem, the Parameters of our SHE protocols, showed in Table 1, are define according to (Pisa, et al., 2012) as: $\rho = \lambda$, $\rho' = 2\lambda$ 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; finally $\tau = \gamma + \lambda + \log_2 b$. Where $O(.)$ is big O notation. In its basic form, the scheme described above is somewhat homomorphic, meaning only a limited number of operations can be performed on encrypted data, before the noise increase makes decryption impossible. In fact this number depends on the magnitude of the noise after every operation. In order to decrypt the correct message, the total noise should not grow more than $p/2$. While after a multiplication we have a significant noise increase, the addition is less problematic because it produces only a slight increase. Estimating the maximum number of possible multiplications is possible by finding the largest μ that satisfies $R^{\mu} < p/2$, where $R = b^{2\lambda+1} + \lambda^5 b^{\lambda}(b^2 + b)$ is the maximum allowed noise.

Similarly we can obtain the limit for the maximum number of additions allowed by the scheme. 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 quite high. For the security basis of the scheme we refer to (Pisa, et al., 2012). As is possible to see from the parameters settings, Table 2, public key is very expensive in terms of memory storage. To provide sufficient security, we need large λ's but this implies big public keys and memory problems. This had led to problems during implementation.

Table 2: Ciphertext, and keys size.

Size of the ciphertext, secret and public key				
λ	Base	Secret Key	Public Key	Cipher text
10	2	10Byte	130 KB	0,1 KB
	2^{10}	101 Byte	1,3 MB	1 KB
	2^{50}	506 Byte	7 MB	6 KB
15	2	25 Byte	7 MB	7 KB
	2^{10}	245 Byte	70 MB	72 KB
	2^{50}	1 KB	350 MB	360 KB
20	2	45 Byte	216 MB	0,1 MB
	2^{10}	451 Byte	2 GB	0,6 MB
	2^{50}	2 KB	10 GB	3 MB

3.Protocol

This section shows how privacy preserving biometric authentication protocols can benefit from the SHE cryptosystem working on integers. We consider that the server has a database containing the biometric templates of *N* members, each represented as an array of features**.**

The protocol is divided into three parts (Figure 1):

- Part 1 (input encryption): The client encrypts its biometric template with his/her public key (pk) and sends it to the server. We assume that the server already has client's pk.
- Part 2 (protocol execution): The server loads the biometric database and encrypts the values necessary for the computation. For each biometry in the database, it evaluates the distance from the probe, in a privacy preserving way as described in Section 3.1**,** and subtracts the acceptance threshold from each result. Distance computation can be parallelized to improve the runtime. We used 4 threads, according to our system characteristics. Then the vector containing all the matching results is randomly permuted, before finally being sent back to the client. This is done to prevent the user obtaining the position of the biometry in the database.
- Part 3 (output decryption): The client decrypts all the elements of the vector with his secret key. If one of the obtained values is a negative integer its biometry matches with an element of the database.

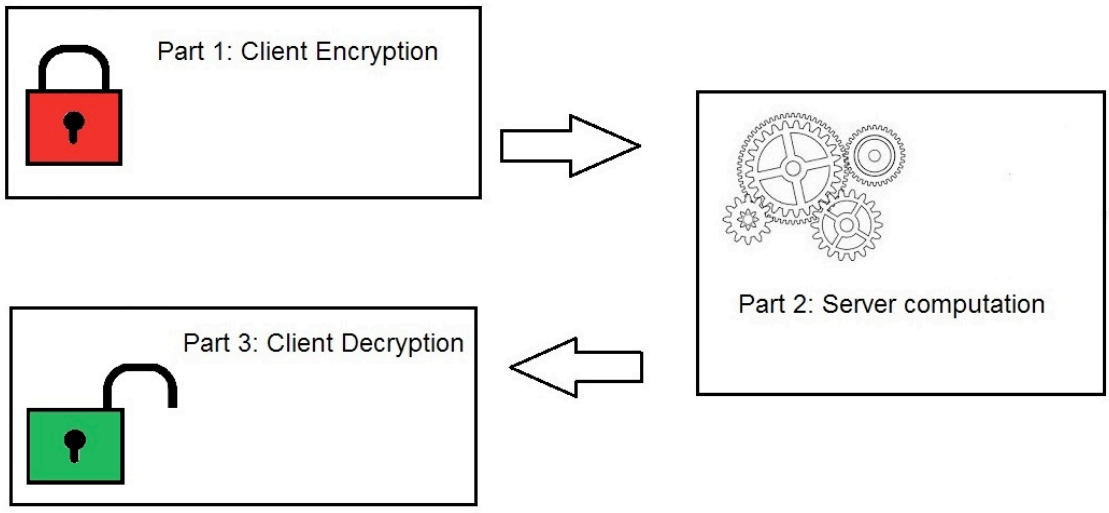

Figure 1: Three parts of the protocol.

3.1 *Distance and thresholding implementation.*

In this section we present our solution to evaluate distances in encrypted domain.

Each feature is a vector of n elements, with $q = (q_1, \dots q_n)$ we indicate the client probe and with

$X_i = (x_{i,1}, \dots x_{i,n})$ the i-th biometry in the database. Furthermore $\oplus$ identifies the XOR and $\| \ \|$ the norm of the binary vector.
If the features are binary values, the Hamming distance is evaluated as $D(q, X_i) = \|(q \oplus X_i)\|$; otherwise the squared Euclidean distance $D(q, X_i) = \|(q - X_i)^2\|$ is computed.
To calculate the Hamming distance, we would need to evaluate the XOR between ciphertexts. However we can perform additions modulus b, so XOR can be evaluated as $(q_j \oplus x_{i,j}) = q_i + x_{i,j} - 2q_j x_{i,j}$. The server has access to the bits of the current biometry and it is possible to avoid computing the multiplication. Hence given the encrypted bits of the probe, server can calculate the XOR as

$$[\![q_i \oplus x_{i,j}]\!] = \begin{cases} [\![q_i]\!] \text{ if } x_{i,j} = 0 \\ [\![1 - q_i]\!] = [\![1]\!] + [\![-1]\!][\![q_i]\!] \text{ if } x_{i,j} = 1 \end{cases}$$

where $[\![\ ]\!]$ denotes the encryption with the user's public key. We again point out that the representation of negative numbers is also possible, as described in Section 2.1.
Given the XOR implementation, the server computes the encrypted Hamming distance as the encrypted sum of all the XORs. Similarly, to compute the Euclidean distance the server performs $\sum_{i=1}^{n}([\![q_j]\!] + [\![-x_{i,j}]\!])^2$. This solution allows us to store only two encrypted numbers for iris (1 and -1) and $b - 1$ values for Euclidean distance, avoiding keeping all database encrypted with client public key.
Now the distance can be compared with the threshold. The comparison operation is not allowed in encrypted domain so instead the server computes the difference between each distance and ε. Two biometrics match if the result is negative. Finally the server randomly permutes the vector and then sends results back to client.

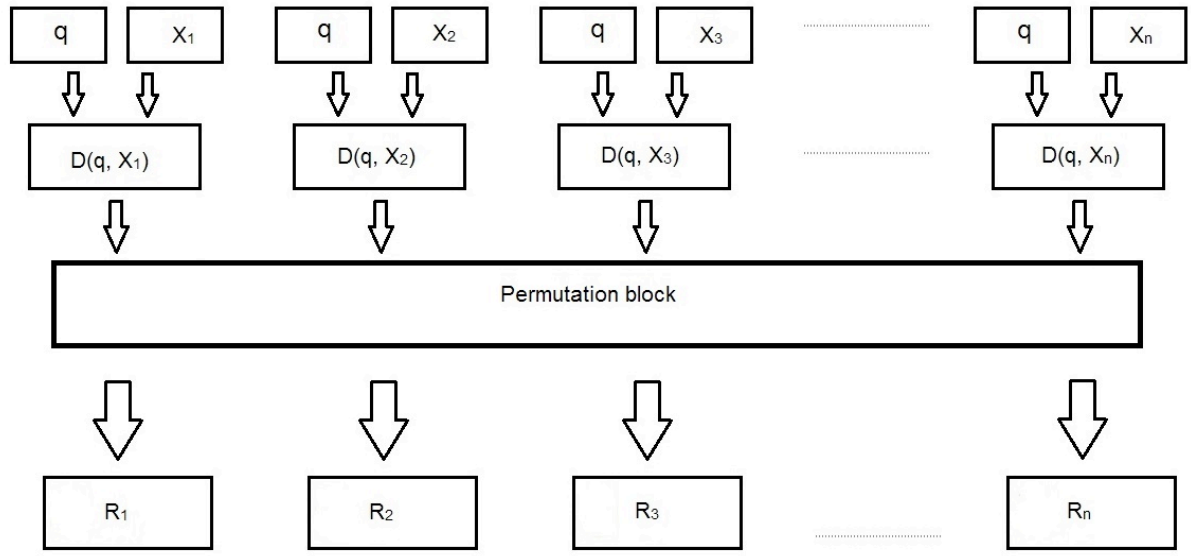

Figure 2: Part 2, server computation.

3.2 *Computational complexity.*
The advantage of our chosen cryptosystem is having the possibility to directly evaluate the Hamming or Euclidean distance on the server, working on encrypted integers.
Indeed the protocol could be computed by using the DGHV scheme with base 2, but while the XOR among the bits could be implemented without products, differences among features, square values and the sum of the feature differences require circuits composed by many AND gates. In the case of the Hamming distance, by using a reverse tree structure to compute the sums, we can observe that the tree is composed by $\log_2 n$ layers, wherein the i-th layer is composed by $n/2^i$ adders of i-bit long inputs (each one needing i AND gates). Hence for each biometry in the database, $2(n-1) - \log_2 n$ 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 a decryption error. More AND gates are needed for the Euclidean distance. On the contrary, since our solution works directly on integer values it requires, on average, only $n/2$ products to compute the XOR between q_i and $x_{i,j}$ (products

can be computed in parallel). With the Euclidean distance, n products are evaluated in parallel to compute the square of $q_i - x_{i,j}$. As a result, we only require that the SHE can cope with the noise amplification due to one multiplication. As said before, it is important to note that for distance computation the server does not need to encrypt the entire database, but only few numbers. In particular - for the case of a Euclidean distance - by assuming that each feature is represented with l bits, it can be more convenient to encrypt the $2^l - 1$ values that the features can assume (encryption of 0 is avoided), at the beginning of the protocol, and select the correct cipher text given the real value of the feature, rather than encrypting all the single features for every database entries, if $2^l - 1 < n \times N$.

4. Experimental validation.

In this section we discuss the results of a practical implementation of the proposed cryptosystem when applied to iris and fingerprint matching.

First of all we are going to analyze the communication complexity of the cryptosystem. According to Section 2, the secret key is in the range $[b^{\eta-1}, b^{\eta})$, with $b = 2^k$, hence, is represented with $k\eta = k\lambda^2$ bits. Similarly we obtain that the secret key and each element composing the public key are represented by up to $k\lambda^5$ bits.

The computational complexity of the scheme has been evaluated in terms of protocol runtime. More specifically, we have measured the average runtimes required by single encryption, decryption, key generation and multiplication, by using a Java implementation of the SHE scheme.

We ran 150 tests on a desktop equipped with a Quad-Core CPU (Intel i7 at 3,40GHz) and 16 GB RAM, mounting a 64-bit Windows OS. The averaged results are shown in Table 3. Addition and secret key generation runtimes are not shown, since they are negligible. The runtime is not significantly affected by the different values assigned to the base, hence our solution of working on integers is preferable to a SHE-based protocol working on bits, thanks to the reduced number of multiplications.

Table 3: Cryptosystem execution times.

Crypto system execution times					
λ	Base	Generation public key	Encryption	Decryption	Multiplication
10	2	0,01 s	0,20 ms	0,00 ms	0,00 ms
	2^{10}	0,04 s	0,17 ms	0,03 ms	0,00 ms
	2^{50}	0,58 s	0,27 ms	1,00 s	3,00 ms
15	2	0,18 s	0,60 ms	0,20 ms	4,67 ms
	2^{10}	3,30 s	2,90 ms	0,01 s	0,10 s
	2^{50}	3 min	0,02 s	0,19 s	1,11 s

For iris matching, we considered the algorithm proposed in (Daugman, 2004), where an iris is represented through a vector of 2048 binary features. Hence the Hamming distance is used and only the encryption of values 1 and -1 is performed on the server side, avoiding the problem of encrypting and storing many features. The maximum value the distance can assume is 2048 and it is represented by 11 bits. For this reason, and to allow for negative numbers, we chose as base $b = 2^{12}$. High values of λ should be used, to guarantee a sufficient security level. However, in considering our requirement for a large base, with $\lambda = 20$ we already have a public key of 2 GB, hence for our tests we considered λ equals to 10 and 15, for which public key is about 1,5 MB and 77 MB.

We performed the tests on randomly chosen bit vector to simulate a possible iris database. The test has been repeated 50 times. We expected a long time for execution due to the lengthy time required for each multiplication (as in Table 3).

Table 4: Protocol execution times.

Iris protocol execution times			
λ	Part 1: Client	Part 2: Server	Part 3: Client
10	0,33 s	0,12 s	0,00 s
15	7,10 s	54 s	0,01 s

We measured the average execution time in seconds of each part of the protocol with respect to a database of a single element. Hence the total time has to be multiplied by a factor *N*. As can be seen in Table 4, for $\lambda = 15$ the most expensive part in terms of execution time is server computation, which takes almost a minute, while the initial client encryption of the 2048 elements vector takes some seconds (and the last part is negligible). On the other side, for $\lambda = 10$ the execution of the whole protocol takes less than a second, but is less secure.
The 2048 ciphertexts are transmitted from the client to the server and another one for each biometric match from the server to the client, resulting in the total transmission of 6MB and 346MB for $\lambda = 10$ and 15 respectively.

For fingerprint matching, we implemented the system described in (Barni, et al., 2010). The authors demonstrate that the representation of a fingercode through a vector of 96 features, each represented with 2 bits, increases the equal error rate from 6.7% of the original plain implementation to only 7.6%. Considering that the maximum value the distance can assume is 864, which can be represented with 10 bits, the configuration results are appealing for a privacy preserving SHE implementation. We also underline that to evaluate the Euclidean distance, only the encryptions of -1, -2 and -3 are needed on the server side, again avoiding the problem of encrypting and storing many features. Tests have been run, again to allow negative numbers representation, with base $b = 2^{11}$ and security parameters 10,15 (Table 5). With this set of parameters, public key is respectively 1,4 MB and 84 MB.
As before, for $\lambda = 10$ the protocol takes less than a second, while for the bigger security parameter the most expensive part is the server's computation, which takes about 4 seconds. In this case 96 ciphertexts are transmitted from client to server and one for each biometric in the database from server to client, with a bandwidth of about 264KB ($\lambda = 10$) and 15 MB ($\lambda = 15$).

Table 5: Fingerprint protocol execution times.

Fingerprint protocol execution times			
λ	Part 1: Client	Part 2: Server	Part 3: Client
10	0,02 s	0,01 s	0,00 s
15	0,34 s	3,76 s	0,01 s

For $\lambda = 10$ the time required by our implementation of the privacy preserving iris and fingerprint matching protocol has similar performances based on GC (Luo, et al., 2012) or Paillier HE (Barni, et al., 2010) implementation. On the contrary, as expected for $\lambda = 15$, the time needed by the SHE implementation is by far larger than the execution time of protocols based on Paillier HE (Paillier, 1999) or GC (Yao, 1986). The full-GC implementation of the iris matching in (Luo, et al., 2012) needs less than 1 second, but both circuit garbling and circuit transmission are pre-computed.
Nonetheless, all computation in our protocol is moved to the server side and no interaction is needed, making the protocol appealing for an offline database search. Moreover, running times can be lowered by using powerful servers that allow for parallelization across a greater number of threads. Also, the *N* matching protocols can be evaluated in parallel.
We also underline that our Java implementation of SHE-based private iris and especially fingerprint recognition are faster than the C++ implementation of face recognition of (Troncoso-Pastoriza, et al., 2013) that runs in 12.3 seconds. But iris recognition is working with 2048 binary features and fingerprint is working with 96 features, while face recognition is using 4000 features, each represented with 3 bits. This difference in performance depends on the language used for implementation as well as on the different SHE setup. On the contrary, our implementation is slower than Yasuda's (Yasuda, et al., 2013), but they have implemented a packing method and consider only bit values.

5.Conclusions.
We extended and implemented a recently proposed SHE scheme (Pisa, et al., 2012) working on integer values and tested its performance. We used it to build a privacy preserving biometric matching protocol whose complexity is lower than that of SHE solutions working only on binary values. Even if the protocol is not as efficient as others based on SMPC techniques, such as Garbled Circuits and Homomorphic Encryption, our solution has the advantage of moving all the computation onto the server side, eliminated the necessity of interaction together with the private key owner. We also avoided encrypting the whole database. This was achieved by encrypting only the values that the features can assume, so there is no necessity to encrypt (and store) the entire database with the user's public key before the protocol starts.

We observed from our tests that runtimes of single operations and bitsize of ciphertext and keys are slightly affected by the base, while they greatly increase when increasing the security parameter. We then tested the iris and fingerprint recognition protocol observing that a single match requires some seconds.
In the future, we are interested to improve the protocol by using some of the innovative solutions on SHE schemes that are proposed everyday. Also, we are keen 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.

Acknowledgements.
I take this opportunity to express gratitude to my PhD supervisor Mauro Barni and to Riccardo Lazzeretti for their help and support.

References

Barni, M., et al., (2010) *A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates*, Fourth IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS), pp. 1-7.
Barni, M. et al., (2010) *Privacy-preserving fingercode authentication*, Proceedings of the 12th ACM workshop on Multimedia and security, pp. 231-240.
Bianchi, T., Piva, A. and Barni, M., (2011) *Analysis of the security of linear blinding techniques from an information theoretical point of view*, In: Acoustic, Seech and Signal Processing (ICASSP),2011 IEEE International Conference on. s.l.:IEEE, pp. 5852-5855.
Blanton, M, and Gasti, P. (2011). *Secure and efficient protocols for iris and fingerprint identification,* Computer Security–ESORICS 2011, pp. 190-209. Springer Berlin Heidelberg.
Brakerski, Z. and Vaikuntanathan, V., (2011) *Fully homomorphic encryption from ring-LWE and security for key dependent messages*, Advances in Cryptology–CRYPTO 2011*,* pp. 505-524.
Brakerski, Z. and Vaikuntanathan, V., (2014) *Efficient fully homomorphic encryption from (standard) LWE*, SIAM Journal on Computing , 43(2), pp. 831-871.
Bringer, J., Chabanne, H., and Patey, A. (2013). *Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends*, Signal Processing Magazine, IEEE, 30(2), 42-52.
Bringer, J., Chabanne, H., Favre, M., Patey, A., Schneider, T., & Zohner, M. (2014). *GSHADE: faster privacy-preserving distance computation and biometric identification,* in Proceedings of the 2nd ACM workshop on Information hiding and multimedia security pp. 187-198. ACM.
Campisi, P., (2013) *Security and Privacy in Biometrics,* s.l.:Springer.
Cheon, J. H., et al. (2013) *Batch Fully Homomorphic Encryption over the Integers*, *EUROCRYPT,* Volume 7881, pp. 315-335.
Cohn, H. and Heninge, N., (2011) *Approximate common divisors via lattices,* Available at: arXiv preprint arXiv:1108.2714
Coron, J.S., Mandal, A., Naccache, D. and Tibouchi, M., (2011), *Fully homomorphic encryption over the integers with shorter public keys*, Advances in Cryptology–CRYPTO 2011*,* pp. 487-504.
Coron, J.S., Naccache, D. and Tibouchi, M., (2012) *Public key compression and modulus switching for fully homomorphic encryption over the integers*, Advances in Cryptology–EUROCRYPT 2012, pp. 446-464.
Daugman, J., (2004) How *Iris Recognition Works*, IEEE Transactions on Circuits and Systems for Video Technology*,* 4(1), pp. 21-30.
Erkin, Z. et al., (2009) *Privacy-preserving face recognition*, Privacy Enhancing Technologies*,* pp. 235-253.
Fontaine, C. and Galand, F.,(2007) *A survey of homomorphic encryption for nonspecialist,* EURASIP Juornal on Information Security.
Gentry, C. and Halevi, S., (2011) *Implementing gentry's fully-homomorphic encryption scheme*, Advances in Cryptology–EUROCRYPT 2011., pp. 129-148.
Gentry C., (2009) *A fully homomorphic encryption scheme*. PhD Thesis. Stanford University.
Goldreich, O., (1998) *Secure multi-party computation,* Manuscript. Preliminary.
Jain, A. et al. (2000), *Filterbank-based fingerprint matching,* Image Processing, IEEE Transactions on, vol.9, no. 5, pp 846-859.

Luo, Y. et al., (2012) *An efficient protocol for private iris-code matching by means of garbled circuits*, 2012 19th IEEE International Conference on Image Processing (ICIP), pp. 2653-2656.
Paillier, P., (1999). *Public-key cryptosystems based on composite degree residuosity classes,* In Advances in cryptology—EUROCRYPT'99 pp. 223-238. Springer Berlin Heidelberg.
Pisa, P. S., Abdalla, M. and Car, O., (2012). *Somewhat homomorphic encryption scheme for arithmetic operations on large integers*, Global Information Infrastructure and Networking Symposium (GIIS), 2012, pp. 1-8.
Prabhakaran, M. M. and Sahai, A., (2013) *Secure Multi-Party Computation.* IOS press.
Rabin, M. O. (1981). *How To Exchange Secrets with Oblivious Transfer*, Technical Report TR-81, Aiken Computation Laboratory, Harvard University, Tech. Rep.
Sadeghi, A.R, Schneider,T. and Wehrenberg,I. (2010). *Efficient privacy-preserving face recognition*, *Information, Security and Cryptology–ICISC 2009*, pp. 229-244. Springer Berlin Heidelberg.
Troncoso-Pastoriza, J. R., I Gonzalez-Jimenez, D. and Perez-Gonzalez, F., (2013) *Fully Private Noninteractive Face Verification*, Information Forensics and Security, IEEE Transactions on , 8(7), pp. 1101-1114.
Van Dijk, M., Gentry, C., Halevi, S. et al. (2010). *Fully homomorphic encryption over the integers,* Advances in Cryptology–EUROCRYPT 2010*,* pp. 24-43.
Yao, A. C.-C., (1986) *How to generate and exchange secrets*, Foundations of Computer Science, 1986,27th Annual Symposium on, pp. 162-167.
Yasuda, M. et al., (2013) *Packed Homomorphic Encryption Based on Ideal Lattices and Its Application to Biometrics*, Security Engineering and Intelligence Informatics*,* pp. 55-74.