Secure Hamming Distance based Biometric Authentication

Rohan Kulkarni Anoop Namboodiri
International Institute of Information Technology
Hyderabad
{rohan.kulkarni@research, anoop@}.iiit.ac.in

Abstract

Concerns of privacy, template security and efficiency of biometric authentication systems have received considerable research attention in the recent past. Binary template representations have been shown to provide significant improvements in efficiency without compromising accuracy for a variety of biometric modalities including fingerprints, palmprints and iris. Most of the secure authentication protocols work with generic feature representations or propose new secure templates for specific modalities. In this work, we propose an authentication protocol that works with any binary biometric representation that employs hamming distance for comparison and convert it into a secure, privacy preserving protocol that provides template protection. We also provide the ability to use masks while computing the hamming distance. The protocol retains the accuracy of the underlying representation as demonstrated by our experiments.

1. Introduction

Biometry and Security go hand in hand in user authentication or identification. The uniqueness of the biometric data of a user provides the credibility of the individual to be authenticated. Thus it provides a way to ensure secure access to an environment. The non-revocability quality of biometrics causes high security to be ensured for the stored data. If stolen, the user's identity in any system which authenticates based on that biometric is in danger. Ensuring perfection in combining both the aspects of security and credibility i.e accuracy for building a system efficient enough to work in real time has not yet reached a satisfactory stage. This is the source of research motivation in the field of biometric authentication[19].

In this paper, we design and implement a system which provides biometric template protection along with a secure authentication mechanism. The proposed protocol requires the underlying matching algorithm to be based on normalized hamming distance of two binary feature vectors. It also allows the use of masks while computing the distance. It's a two party protocol with the server having the secure biometric templates and the client attempting the authentication using its own biometric data along with some keys. The authentication process takes place on encrypted data and does not allow any leakage of information about the biometric features.

State-of-the-art cryptographic protocols are not designed for error-tolerance in their inputs. On the other hand, biometric systems have to be built on a classifier which tolerates some amount of fuzziness in its data. Thus, combining biometrics and cryptographic protocols to develop a secure system is a difficult problem. It can be dealt in two ways, either develop a stable feature from biometric data or make the matching algorithm a part of the protocol. However, both the ways are quite hard.

Most of the systems which try to integrate the merits of both, cryptography and biometrics, use Fuzzy Extractors or Secure Sketches[15, 11]. The underlying mechanism uses error correction codes to handle the fuzziness in the biometric data. The limitations of correction capacity affect the matching accuracy of the system. Many hashing based systems have also been developed but they too fail to achieve high accuracy in realistic settings[16].

Several biometric matching systems are based on strong cryptographic primitives - the homomorphic Paillier[17], Goldwasser-Micali cryptosystem[8, 5], Garbled circuits[14, 9], [3] which propose secure methods for user authentication, but are not equipped to provide security, privacy and template protection simultaneously with efficient computation. Of them, [14, 8] provide secure identification, [3, 9] even involve a mask vector, but do not ensure template protection. [5, 17] are based on secure hamming distances, however there is no mask vector involved.

The protocols in [6, 7] operate on an encrypted domain and perform biometric identification. Their computational complexities for performing a matching are high for a real time authentication system. A secure and private authentication system is proposed in [20]. It is designed for biometric schemes which use linear classifiers and SVMs.

Biometric systems for fingerprints, iris as well as palmprints exist which are based on hamming distance as the dissimilarity measure. We limit the description of our protocol to iris matching based on *IrisCodes*, but conduct experiments on iris and palmprint matching. The palmprint matching method, based on *PalmCodes*, is similar to the iris matching technique. The proposed protocol ensures privacy, security, fixed rounds of communications for the authentication keeping intact the accuracy provided by the underlying matching algorithm.

2. Preliminaries

In this section we give an overview of the algorithms used by our protocol. We first describe the encryption mechanism and supported homomorphic operations. Then we briefly describe the Iris matching technique used.

2.1. Encryption scheme:

We use the *somewhat* homomorphic encryption scheme proposed by Boneh *et al.* [4] constructed on the lines of the Paillier's encryption scheme[18]. It allows additive homomorphism of ciphertexts, also supports *one* multiplication operation between the ciphertexts. The cryptosystem is based on finite composite order groups built on elliptic curves that support bilinear maps. Its security depends on the hardness of the *subgroup decision problem* - the problem of determining whether a given element of a finite group G lies in a specified proper subgroup G_1 or not. Then for composite order groups, say of order $n = p_1p_2$, the hardness of the subgroup decision problem directly transforms to the hardness of factoring n. High security demands that it must be infeasible to factor it, so n must be considerably large. The drawback is - computing group operations and pairings on large composite order groups is prohibitively slow. Prime order groups on the other hand can provide equivalent security with a smaller order[2] and allow faster computation of pairings. The subgroup decision problem can be constructed on a prime order group using a random generator and a subgroup. The generator of the subgroup is the group generator raised to a random integer in its field. A Tate pairing on a 1024 *bit* composite order elliptic curve group is roughly 50 times slower than that on a comparable 160 *bit* prime-order curve group[12].

Freeman[12] proposed a system to convert composite-order groups to prime-order that encompasses the bilinear properties and holds the hardness of the subgroup decision problem. It provides a framework for using prime-order elliptic curves to construct bilinear groups to create efficient versions of the cryptosystems that originally used composite-order groups. The generated prime-order groups are equipped with projection-maps that map them onto proper subgroups and commute with the pairing. We implement this particular scheme and use it to build our protocol. The prime-order groups also support the required homomorphic additions of ciphertexts along with *one* multiplication. The algorithms provided by the scheme are described below.

KeyGen(λ): Let $\mathcal{G}$ be a projecting bilinear group generator. Provided a security parameter λ, compute $(G, G_1, H, H_1, G_t, G'_t, e, \pi_1, \pi_2, \pi_t) \leftarrow \mathcal{G}(\lambda)$. Choose $g \xleftarrow{R} G, h \xleftarrow{R} H$ and output the public key $PK = (G, G_1, H, H_1, G_t, G'_t, e, g, h)$ and the secret key $SK = (\pi_1, \pi_2, \pi_t)$. *Where*,

λ is the security parameter, to generate λ bit prime p.

G, G_1, H, H_1, G_t and G'_t are Groups of order p over an elliptic curve such that $G_1 \subset G, H_1 \subset H \; and \; G'_t \subset G_t$.

g, h are random generators of G, H respectively.

$e = G \times H \rightarrow G_t$ is a bilinear map.

$\{\pi_1, \pi_2, \pi_t\}$ are the projection maps of their corresponding groups, trapdoors used in the decryption process.

Encrypt($\mathcal{PK}, m$): To encrypt a message m using a public key, choose $g_1 \xleftarrow{R} G_1$ and $h_1 \xleftarrow{R} H_1$ and output the ciphertexts $(C_g, C_h) = (g^m.g_1, h^m.h_1) \in G \times H$.

Decrypt($\mathcal{SK}, \mathcal{C}$): The input ciphertext C can be an element of $G, H \; or \; G_t$.

- If $C \in G$, output $m \longleftarrow log_{\pi_1(g)}(\pi_1(C))$
- If $C \in H$, output $m \longleftarrow log_{\pi_2(h)}(\pi_2(C))$
- If $C \in G_t$, output $m \longleftarrow log_{\pi_t(e(g,h))}(\pi_t(C))$

Homomorphic properties: The system is additively homomorphic within the respective groups and supports *one* multiplication operation between the groups.

Add($\mathcal{PK}, \mathcal{C}, \mathcal{C}'$): The two ciphertexts C, C' are in one of $G, H \; or \; G_t$. Choose $g_1 \xleftarrow{R} G_1$ and $h_1 \xleftarrow{R} H_1$ and perform:

- If $C, C' \in G$, output $C.C'.g_1 \in G$
- If $C, C' \in H$, output $C.C'.h_1 \in H$
- If $C, C' \in G_t$, output $C.C'.e(g, h_1).e(g_1, h) \in G_t$

We use $+_E$ symbol for this operation in our protocol.

Multiply($\mathcal{PK}, \mathcal{C}, \mathcal{C}'$): The two ciphertexts are - $C \in G$ and $C' \in H$. Choose $g_1 \xleftarrow{R} G_1$ and $h_1 \xleftarrow{R} H_1$ and output $C = e(C, C').e(g, h_1).e(g_1, h) \in G_t$

We use $*_E$ symbol for this operation in our protocol.

2.2. Iris Matching

Let B denote the iris feature vector, a k-bit binary array. The feature extraction algorithms [21, 10] compute this bit vector from a scanned iris image. It involves applying various Gabor filters on separate local areas of the iris image to generate an *IrisCode*. The feature extraction algorithm, along with the feature vector, outputs the reliable and unreliable bits in it. This is provided through a binary mask vector of same length, with a '1' at the location which is valid or reliable and '0' at the position where the value in the feature vector should be discarded. So, provided two biometric

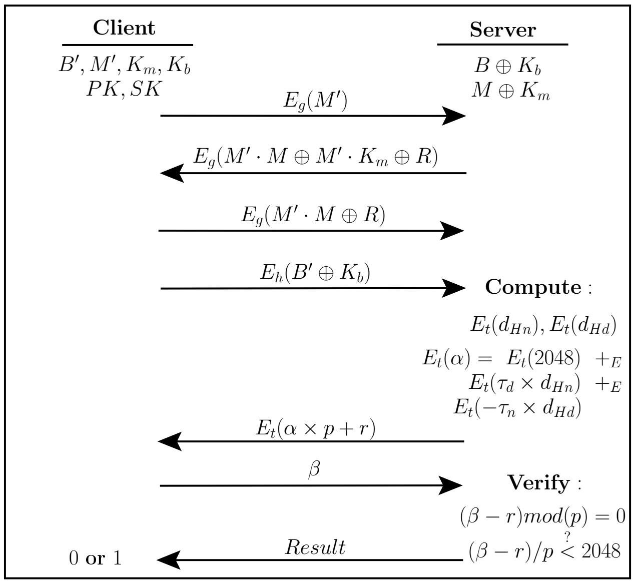
Client
Server
B', M', K_m, K_b
PK, SK
$B \oplus K_b$
$M \oplus K_m$
$E_g(M')$
$E_g(M' \cdot M \oplus M' \cdot K_m \oplus R)$
$E_g(M' \cdot M \oplus R)$
$E_h(B' \oplus K_b)$
Compute :
$E_t(d_{Hn}), E_t(d_{Hd})$
$E_t(\alpha) = E_t(2048) +_E$
$E_t(\tau_d \times d_{Hn}) +_E$
$E_t(-\tau_n \times d_{Hd})$
$E_t(\alpha \times p + r)$
β
Verify :
$(\beta - r) mod(p) = 0$
$(\beta - r)/p \overset{?}{<} 2048$
0 or 1
Result

Figure 1. Working of the protocol

feature vectors B' and B and their corresponding masks M' and M, the normalized hamming distance between them is used as a measure of dissimilarity between the two samples. The normalized hamming distance is given by

$$d_H(B', M', B, M) = \frac{||(B' \oplus B) \cdot M' \cdot M||}{||M' \cdot M||} \qquad (1)$$

The distances are calculated over some rotations of the second template as the two images can be slightly mis-aligned. By considering the minimum of the distances calculated, the minor orientation errors are compensated. A threshold value τ is chosen upon by the matching algorithm and based on the comparison of d_H with τ, the result is a match or a mis-match. This τ is based on the distributions of authentic and imposter data. It is selected in a way to achieve the desired FAR and GRR.

PalmCode based palmprint matching is very similar, also uses the same dissimilarity measure[22]. We perform our experiments on these two biometric modalities.

3. Proposed Protocol

The proposed system is a two phase protocol, built over a *client-server* architecture. Initially, the user registers himself on the server, whose access he seeks, by submitting his biometric data to it. This is the enrollment phase. The user extracts and sends its biometric features, in our case the iris template(feature vector and mask) to a trusted enrollment server. The server produces two binary keys, one for the iris feature vector and the other for the mask. The values stored on the server are the XORed values of the template vector with the corresponding key. Let B be the binary iris feature vector and M be its mask provided by the user. The server generates two bit vectors K_b and K_m of lengths equal to that of B and M respectively. The server stores $(B \oplus K_b)$ and $(M \oplus K_m)$ on it, and the keys K_b and K_m are handed over to the user as they will be required while authenticating himself in the future.

In the authentication phase the user, client from now on, has to prove its identity. It gets involved in a protocol with the server for gaining access to the application built on it. *Figure 1* gives the overview of the protocol. We describe it below in detail.

Authentication phase: The client extracts the feature vector B', mask M' from its iris biometric. It has two keys, K_b and K_m. It generates a pair of public and private keys of the above mentioned cryptosystem. The public key $PK = (G, G_1, H, H_1, G_t, G'_t, e, g, h)$ and the secret key $SK = (\pi_1, \pi_2, \pi_t)$. The client publishes his public key to the server and then follows the protocol in *Figure 2*.

INPUT: The client's input is - biometric vector B', biometric-key K_b, mask M' and mask-key K_m each $\in \{0,1\}^l$. The server has the enrolled biometric templates $B \oplus K_b, M \oplus K_m$ of all the users registered on it and a threshold $\tau = \tau_n/\tau_d$.

OUTPUT: The server learns α(matching score) of the client's biometric template and then communicates the authentication result to the client.

PROTOCOL:

1. The client performs the G group encryption on each bit of M' and sends it to the server along with it's identity.
2. The server receives $ID, \{E_g(m'_0), \ldots, E_g(m'_{l-1})\}$, generates a random binary vector $R = \{r_0, r_1, \ldots, r_{l-1}\}$ and performs $\forall i, (E_g(m'_i) \cdot (m_i \oplus k_{mi})) \oplus r_i$ to get the following values:

   $\forall i, (E_g(m'_i \cdot m_i \oplus m'_i \cdot k_{mi} \oplus r_i))$. It sends them over to the client. (Efficient ways of calculating the $\oplus$ and $\cdot$ operations of an encrypted value with a plain value are mentioned in the text)
3. The client decrypts the binary vector and eliminates the term involving the mask-key.

   $\forall i, (m'_i \cdot m_i \oplus m'_i \cdot k_{mi} \oplus r_i) \oplus (m'_i \cdot k_{mi}) = (m'_i \cdot m_i \oplus r_i)$

   It again performs G group encryption of the result and sends it to the server.
4. The server now receives $\{E_g(m'_0 \cdot m_0 \oplus r_0), \ldots, E_g(m'_{l-1} \cdot m_{l-1} \oplus r_{l-1})\}$ and calculates $\forall i, E_g(m'_i \cdot m_i \oplus r_i) \oplus r_i$ to get the final mask vector $\{E_g(m'_0 \cdot m_0), \ldots, E_g(m'_{l-1} \cdot m_{l-1})\}$
5. The client calculates $B' \oplus K_b$ and sends the H group encryption of its each bit to the server.

6. The server receives the encrypted bits and calculates $\forall i, E_h(b'_i \oplus k_{bi}) \oplus (b_i \oplus k_{bi})$ to get $\{E_h(b'_0 \oplus b_0), \ldots, E_h(b'_{l-1} \oplus b_{l-1})\}$

   Now, the server calculates

   $\forall i, E_t((b'_i \oplus b_i) \cdot m'_i \cdot m_i) = E_h(b'_i \oplus b_i) *_E E_g(m'_i \cdot m_i)$.

   It then calculates,

   $E_t(d_{Hn}) = +_E{}_{i=0}^{l-1} E_t((b'_i \oplus b_i) \cdot m'_i \cdot m_i)$

   $E_g(d_{Hd}) = +_E{}_{i=0}^{l-1} E_g(m'_i \cdot m_i)$
7. It then calculates $E_t(\tau_d \times d_{Hn}),\ E_g(\tau_n \times d_{Hd})$ using $+_E$ and performs $E_g(\tau_n \times d_{Hd}) *_E E_h(1)$ to obtain $E_t(\tau_n \times d_{Hd})$.

   It then computes $E_t(\alpha) = E_t(2048 + \tau_d \times d_{Hn} - \tau_n \times d_{Hd})$ using the $+_E$.

   Where, α will be in the range $[0, 4096]$. The server then randomizes it - $E_t(\alpha \times p + r)$ and sends it to the client.
8. The client decrypts the ciphertext and sends back the plaintext. Let it be β.
9. The server receives β and performs the de-randomization with a check. The server checks if $(\beta - r) mod(p) = 0$. If it holds then it sends the authentication result:

   $$Result = \begin{cases} 1 & \textit{If } (\beta - r)/p < 2048 \\ 0 & \textit{otherwise} \end{cases}$$

Figure 2: Description of the protocol

The client performs the encryptions as mentioned in Section 2.1. $E_g, E_h \ and \ E_t$ denote $G, H \ and \ G_t$ group encryptions, which are additively homomorphic in their respective groups. A multiplication operation performed between two ciphertexts, one belonging to G and the other belonging to H outputs a ciphertext belonging to G_t.

The client encrypts the bits of mask M' with E_g, whereas that of B' with E_h. The server performs the required XOR and AND operations on individual group elements with known plain text bits as in *Steps 2, 4* and *6* of the protocol. Let $E(b'_i)$ be an encrypted bit, at location i of some vector available with the server. To compute a '$\oplus$' operation with the server's own bit b_i, the server keeps $E(b'_i)$ as the result if $b_i = 0$ and computes the result as $E(-b'_i) + E(1)$ if $b_i = 1$. The negative of the encrypted value is calculated using its group inverse. For a '$\cdot$' operation with the server's own bit b_i, it considers 0 as the result if $b_i = 0$ and keeps $E(b'_i)$ as the result if $b_i = 1$. The server then blinds the result adding encryption of a random value $E(v)$ from the corresponding subgroup. $v \overset{R}{\longleftarrow} G_1$ and E_g encryption is used if $E(b'_i) \in G$ else $v \overset{R}{\longleftarrow} H_1$ and E_h encryption is used if $E(b'_i) \in H$.

The multiplication operation in *Step 6* is performed between the encrypted bits of $m'_i \cdot m_i$ and $b'_i \oplus b_i$ which belong to G and H respectively. The server then adds up the corresponding encrypted bits to obtain $E_t(d_{Hn}) \ and \ E_g(d_{Hd})$ - encrypted numerator and denominator of the normalized hamming distance.

The threshold(τ) is represented as a fraction(τ_n/τ_d) as only integer operations are supported by the cryptosystem. In *Step 7*, the additive homomorphism allows computation of $E_t(\tau_d \times d_{Hn})$ and $E_g(\tau_n \times d_{Hd})$. Next, the homomorphic multiply with $E_h(1)$ converts $E_g(\tau_n \times d_{Hd})$ to a G_t group encryption. Then the server computes the value $E_t(\tau_d \times d_{Hn} - \tau_n \times d_{Hd})$ using the additive homomorphic properties of E_t. Now, to decide the authentication result it needs to know if the encrypted value is *positive* or *negative*. The server adds 2048 to the encrypted value to keep it in a non-negative domain helping the decryption, as described in *Section 4.2*. To hide the value from the client the server also randomizes it.

4. Protocol Analysis

In this section we perform a detailed analysis of the proposed protocol with regard to its correctness, computability and security.

4.1. Correctness

The cryptosystem provides projection functions - $\{\pi_1, \pi_2, \pi_t\}$, also called trapdoors, when applied on the ciphertexts eliminate the random values added while performing mathematical operations. Computing a discrete logarithm of those with the base set to the projection values of the corresponding group generators successfully outputs the encrypted message.

The correctness of the protocol also depends on the following binary bit operations that hold true:

$(b \oplus b') \oplus b = b' \quad ; \quad a \cdot (b \oplus b') = (a \cdot b) \oplus (a \cdot b')$

They support the template protection scheme used and the randomization of the mask vector at *Step 3*.

4.2. Computational Analysis

The encryption-decryption, homomorphic additive and multiplicative operations are the expensive ones among the total computations performed during the complete execution of the protocol. Optimizing them in any way can reduce the live authentication time of the protocol. We suggest offline computations and optimizations in order to reduce the online time taken by the system. The encryption operations can be pushed offline. The client can pre-compute sufficient number of encryptions of $0's$ and $1's$ before the start of the process. This reduces the encryption time in the online setting to $\mathcal{O}(1)$. The homomorphic additions require negligible time compared to the bilinear pairings in the multiplication operation. The pairings can't be pushed offline or pre-computed as both its inputs are ciphertexts, available only after *Step 6* of the protocol. The number of pairings to be computed can be reduced.

As defined in *Section 2.1*, 3 pairings are computed per multiplication operation where $G_g \times G_h \rightarrow G_t$ action is performed and 2 pairings are computed per addition for a ciphertext $C \in G_t$. The actual multiplication operation requires one pairing , the rest are for randomizing the output. Similarly, the actual addition operation in G_t does not require any pairing. The server computes these operations in *Step 6* of the protocol. We reduce the computation to just one pairing per operation and, in the end, perform a single randomization of the ciphertext.

The decryption operation requires computing discrete logarithms. Computing discrete logarithm of any Integer has a complexity of $\mathcal{O}(\sqrt{s})$ using the Pollard's *rho* algorithm; where s is the search space size. The client needs to compute the discrete logarithms of the ciphertexts received in *Step 3* and *Step 7* of the protocol. We propose the client to perform pre-computations of the exponents using its private key to reduce the complexity to $\mathcal{O}(1)$. The number of pre-computations required depend on the range of p and r chosen by the server. We propose to limit the values to $\mathcal{O}(10^3)$ causing the value '$\alpha \times p + r$' to be $\mathcal{O}(4.097 \times 10^6)$. Keeping it in a non-negative and definite range allows successful decryption using the pre-computations. Storing pre-computed values by hashing allows quick decryption of the randomized distances.

4.3. Security Analysis

Security of a biometric system depends on its design and the security of the underlying cryptographic schemes used. Its design may provide various points of attack for an adversary without valid credentials. The privacy offered depends on the data revealed while undergoing the authentication protocol. The protocol that we describe is secure against a semi-honest adversary. A client which deviates from the prescribed steps can't learn any information about the biometric template at the server. We show that the chances of such deviations leading to acceptance are minimal.

Server security: Let us assume that the adversary gains access to the server database. The templates present in the server database are all XORed with the keys generated by the enrollment server. They do not reveal any information about the biometric of the user. If a user is registered in different authentication servers which use this protocol, the keys used in them will be different, leaking no information even when multiple templates are gathered breaking into multiple servers. If some template is suspected to be broken, a new one with a different key can be generated.

During an authentication process, all the data received from the client is encrypted except the final 'β'. The protocol will not reveal any information about the client's biometric B' or M' except the matching score.

Client-End security: An adversary having access to the client's system cannot carry out authentication without access to both the biometric and the key. If the client tries blind attacks, the amount of effort required is equal to randomly guessing the bit vector. If the client modifies the bits to be sent to the server hoping to learn some information of the biometric present at the server, the randomization in *Step 2* will blind the server vectors from the client. In *Step 7* of the protocol, the client learns the randomized value '$\alpha \times p + r$'. He can send a modified value to pass the authentication test. However, the probability of successfully modifying it to pass the server divisibility test is small.

Network security: An adversary sniffing over the network can only access the encrypted biometric data or data computed using the random numbers generated by the server. This will not help in deciphering any information about the biometric templates, also defend a replay attack.

5. Experiments

5.1. Implementation and Results

For evaluation purpose, the protocol was implemented on a *client-server* architecture in C++ using the MIRACL library. The experiments were performed on a single core of an Intel $3.2Ghz$ processor with around 200 KB of memory usage. The public ICE 2005 database [1] was chosen which has 2953 iris images from 243 eyes. For palmprints, the PolyU database consisting of 7752 images from 386 people was chosen. The homomorphic algorithm was implemented using the type-3 pairing API provided by the library. The pairing based groups were generated for two variants of curves: 1. MNT curve which provides security equivalent to an 80 bit AES encryption and, 2. BN curve which provides security equivalent to an 128 bit AES encryption. The *IrisCodes* and *PalmCodes*, extracted using the algorithms described in *section 2.2*, both consist of a 2048 bit binary feature vector and a same size mask vector. To verify the template matching performance of our proposed system, we compared our results by performing the matching on plain text *IrisCodes* and *PalmCodes*. Both produced the same GAR and FRR values.

All pre-computations and encryptions mentioned in *Section 4.2* were implemented offline. The results are mentioned in *Table 1*. *IrisCodes* and *PalmCodes*, both are same size vectors so their matching times are same. The server values are split into time for bilinear pairings in the multiplications and time for rest of the operations. These server computations are inherently parallelizable and one can achieve near-linear speedup w.r.t. number of cores.

We compare our results with a recent efficient implemen-

Curve	*Security*	*Client*	*Server*			*Bandwidth*
			Pairing	*Rest*	*Total*	
MNT	$80\ bits$	$10\ ms$	$57.1\ s$	$0.9\ s$	$58\ s$	$400\ KB$
BN	$128\ bits$	$10\ ms$	$90.6\ s$	$1.4\ s$	$92\ s$	$640\ KB$

Table 1. Security, computation and bandwidth requirement with different pairing schemes.

Scheme	*Time*	*Bandwidth*	*Comm. Rounds*
[20]	$4\ min$	$5\ MB$	2
Proposed	$58\ sec$	$400\ KB$	3

Table 2. Comparison with existing similar systems.

tation of a secure biometric matching system, which supports template protection in *Table 2*. For the scheme in [20], the values are generated on their $1024\ bit$ public encryption, feature vector size of $2048\ bits$. We don't take into account their dedicated hardware accelerations. We compare their results with our $80\ bit$ AES security implementation. To the best of our knowledge, there is no other work which supports template security and also involves a mask vector, providing specific details of computation and communication costs.

5.2. Alternate method

Garbled circuits allow construction of an alternate solution to the addressed problem. [13] provide efficient implementation of the same. We plan to improve the computational efficiency of our protocol by constructing efficient garbled circuits. Also, we plan to extend our protocol to 1:N matching with further optimizations.

6. Conclusion

We propose a novel protocol which is able to achieve secure and private authentication and also provides template protection without loss of accuracy of the matching algorithm. If offers template revocability, if one is suspected to be broken. The user provides his identity only at the time of enrollment. Once the trusted server secures the templates, no information about the user biometric is leaked. We measure the efficiency of the protocol by implementing it on a *client-server* architecture and estimate the computation and communication complexities. We allow linear scaling of the protocol by increasing the number of processors. This permits verification to be performed in real time with help of available hardware and proposed pre-computations.

References

[1] National institute of standards and technology (nist). In *Iris Challenge Evaluation. http://iris.nist.gov/ICE*, 2005.

[2] E. Barker et al. Recommendation for key management–part 1: General (revised). *NIST special publication*, 800:57, 2011.

[3] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. *Computer Security–ESORICS 2011*, pages 190–209, 2011.

[4] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. *Theory of Cryptography*, pages 325–341, 2005.

[5] J. Bringer and H. Chabanne. An authentication protocol with encrypted biometric data. *Progress in Cryptology–AFRICACRYPT 2008*, pages 109–124, 2008.

[6] J. Bringer, H. Chabanne, and B. Kindarji. Error-tolerant searchable encryption. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[7] J. Bringer, H. Chabanne, and B. Kindarji. Identification with encrypted biometric data. *Security and Communication Networks*, 4(5):548–562, 2010.

[8] J. Bringer et al. An application of the goldwasser-micali cryptosystem to biometric authentication. In *Information Security and Privacy*, pages 96–106. Springer, 2007.

[9] J. Bringer, M. Favre, H. Chabanne, and A. Patey. Faster secure computation for biometric identification using filtering. In *Biometrics (ICB), 2012 5th IAPR International Conference on*, pages 257–264. IEEE, 2012.

[10] J. Daugman. The importance of being random: statistical principles of iris recognition. *Pattern recognition*, 36(2):279–291, 2003.

[11] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in cryptology-Eurocrypt 2004*, pages 523–540. Springer, 2004.

[12] D. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. *Advances in Cryptology–EUROCRYPT 2010*, pages 44–61, 2010.

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

[14] Y. Huang, L. Malka, D. Evans, and J. Katz. Efficient privacy-preserving biometric identification. In *Network and Distributed System Security Symposium*, 2011.

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

[16] A. Kong et al. An analysis of biohashing and its variants. *Pattern Recognition*, 39(7):1359–1368, 2006.

[17] M. Osadchy et al. Scifi-a system for secure face identification. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 239–254. IEEE, 2010.

[18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99*, pages 223–238. Springer, 1999.

[19] U. Uludag et al. Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, 2004.

[20] M. Upmanyu et al. Blind authentication: a secure crypto-biometric verification protocol. *Information Forensics and Security, IEEE Transactions on*, 5(2):255–268, 2010.

[21] R. Wildes. Iris recognition: an emerging biometric technology. *Proceedings of the IEEE*, 85(9):1348–1363, 1997.

[22] D. Zhang, W.-K. Kong, J. You, and M. Wong. Online palmprint identification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1041–1050, 2003.