ANONYMOUS BIOMETRIC ACCESS CONTROL BASED ON HOMOMORPHIC ENCRYPTION

Ying Luo, Sen-ching S. Cheung, and Shuiming Ye

Center for Visualization and Virtual Environments
University of Kentucky, Lexington KY 40507

ABSTRACT

In this paper, we consider the problem of incorporating privacy protection in a biometric access control system. An Anonymous Biometric Access Control (ABAC) system is proposed to verify the membership status of a user using biometric signals without knowing his or her true identity. This system is useful in protecting the privacy of authorized users while keeping out potential imposters and attackers. In our proposed system, we employ homomorphic encryption to protect the probe biometric and propose a Secure Similarity Search (SSS) algorithm to authenticate the probe in an anonymous way. We demonstrate our proposed system using a gallery of iris biometrics and show the efficiency of the implemented anonymous biometric matching performed in encrypted domain.

Index Terms— Biometric Access Control, Privacy Protection, Anonymity, Homomorphic Encryption

1. INTRODUCTION

Providing anonymity is one of the central themes of privacy protection. From obfuscating and removing identity information in images [1] to anonymizing emails and postings [2], many systems have been developed to protect the privacy of individuals in a variety of digital mediums. On the other end of the spectrum are the myriad of Biometric Access Control (BAC) systems that are available to control allocation of resources based on the identity of the user. As a biometric signal is based on "who you are" rather than "what you have", BAC systems excel in authenticating a user's identity. The use of BAC systems, however, creates a conundrum for privacy advocates as the knowledge of the identity in the system makes it hard to keep users anonymous.

A moment of thought reveals that many access control systems do not need the true identity of the user but simply require a confirmation that the user is an authorized member. A typical example will include those service providers that provide free or flat-rate services to their clients. For example, an online movie vendor may have a category of "VIP" members who pay a flat monthly membership fee and can enjoy an unlimited number of movie downloads. While it is important to verify the VIP status of a user, it is unnecessary to precisely identify who the user is. In fact, it will be appeasing to customers if the vendor can provide a guarantee that it can never track their movie selections. Another example is a community electronic message board. Only the members of the community can sign in to the system. Once they sign in to the system, they can anonymously post messages and complain to the entire community. Both examples can benefit from an access control system that can verify the membership status using biometric signals while keeping the identity anonymous. This is the primary design goal of the Anonymous Biometric Access Control (ABAC) system proposed in this paper.

In this paper, we introduce an ABAC system based on homomorphic encryption. The probe biometric is protected by homomorphic encryption so that the biometric server cannot directly examine the probe. On the other hand, we devise a hamming distance based Secure Similarity Search (SSS) module so that the biometric server can compute the hamming distance entirely in the encrypted domain between the probe and each of the records in the gallery. With the help of a non-colluding commodity server, the biometric server can compare the computed distances with a pre-defined similarity threshold without leaking any information about the gallery or the probe to the commodity server. The outcome of SSS is a single bit from the commodity server to the biometric server indicating whether the user is a member of the server. While this provides anonymity to the user, it opens door to possible collusion between the commodity server and an imposter. To protect the system against such attacks, multiple rounds of SSS are carried out with the outcomes of some of the rounds secretly controlled by the biometric server. It can be shown that this strategy makes the probability of the success of such attacks arbitrarily small.

The main contributions of this paper are the introduction of the ABAC system concept and a practical design of such a system using iris biometrics. There are other works that deal with the privacy and security issues in biometric systems but their focus are different from this paper. A privacy-protecting technology called "Cancelable Biometrics" has been proposed in [3]. To protect the security of the raw biometric

signals, a cancelable biometric system distorts a biometric signal using a specially designed non-invertible transform so that similarity comparison can still be performed after distortion. Biometric Encryption (BE) described in [4] possesses all the functionality of Cancelable Biometrics, and is immune against the substitution attack because it outputs a key which is securely bound to a biometric. The BE templates stored in the gallery have been shown to protect both the biometrics themselves and the keys. All the above technologies focus on the security and privacy of the biometric signals in the gallery: instead of storing the original biometric signal, they keep only the transformed and noninvertible feature or help data extracted from the original signal that do not compromise the security of the system even if they are stolen. In all these systems, the identity of the user is always recognized by the system after the biometric matching is performed. To the best of our knowledge, there are no other biometric access systems that can provide access control and yet keep the user anonymous.

This paper is organized as follows. Section 2 provides an overview of the proposed ABAC system and proves that the ABAC is privacy-protected and probabilistically secure. Section 3 presents the implementation details of the SSS module. Experiment results and discussions are presented in Section 4. We conclude this paper with prospect for the future work in Section 5.

2. SYSTEM OVERVIEW

In our proposed system, the biometric server, Bob, has a gallery that stores the biometric signals $\{X_1, \ldots, X_N\}$ of N members where X_i is a binary vector denoted as $(x_1^i, \ldots, x_n^i)$. The user, Alice, provides a probe $q = (q_1, \ldots, q_n)$ for logging into the system and produces a match if there exists an unique i where $i \in \{1, \ldots, N\}$ such that $d(q, X_i) < \epsilon$ for a similarity threshold ϵ. While our setting is general for arbitrary biometric signals, we will focus on iris recognition as described in [5]. In this case, $d(q, X_i)$ denotes a modified Hamming distance as described in Section 3. Both Alice and Bob are assumed to be semi-honest and computationally-bound – they follow the protocols faithfully but will try to gain as much information about the other as possible. Homomorphic encryption is denoted as $Enc(x, pk_C)$ where x is the plaintext and pk_C is a well-advertised public key. We also need a commodity server, Charlie, which assists Bob in the matching process and is the only party that possesses the secret key sk_C for decryption. Charlie is assumed to be semi-honest and non-colluding with Bob. However, we do not assume Charlie to be trustworthy with either the probe or the gallery.

A high-level description of our proposed system is given by Algorithm 1. Step 2 of Algorithm 1 is a key component which we call Secure Similarity Search (SSS). Only a brief outline is provided in step 2 and more details can be found in Section 3. As later demonstrated, our SSS step can guarantee: (1) Charlie will not gain any information about the gallery X_i for $i = 1, \ldots, N$, the probe q and the similarity threshold ϵ; (2) Bob will not gain any information about the probe q. At the end of the SSS step, Bob computes $Enc(c_k^i, pk_C)$ where $c_k^i = 0$ for any $k = 1, \ldots, \lceil \log_2 n \rceil$ indicates that $d(q, X_i) < \epsilon'$. As Charlie possesses the secret key sk_C, Bob sends $Enc(c_k^i, pk_C)$ to Charlie for decryption. The sending order is scrambled so that Charlie does not know which user produces a match, if any. In step 4, Charlie computes and sends to Bob $a := \prod_{i,k} c_k^i$ which will be 0 if and only if there is a match. Since Bob only knows the value of a, he cannot identity the record that produces the match. On the other hand, Charlie can help an imposter by seting a to 0. However, if Bob uses $\epsilon' = 0$ in step 2.3, Bob knows that there should not be a match and the reception of $a = 0$ will indicate an attack. Step 5 ensures that an imposter has to guess which ϵ' Bob uses correctly for all M times in order to launch a successful attack. Suppose we want to keep the probability of admitting an imposter to be less than ζ, we simply need $M = \lceil -\log_2(\zeta) \rceil$ rounds.

Algorithm 1 ABAC System

Require: Bob: X_i, $i \in \{1, 2, \ldots, N\}$, ϵ and M; Alice: q; Charlie: pk_C

Ensure: Bob confirms Alice's membership

1: Alice encrypts q and $NOT(q)$ with pk_C bit by bit and sends $Enc(q_i, pk_C)$ and $Enc(NOT(q_i), pk_C)$ to Bob for $i = 1, \ldots, n$
2: SSS: (1) Bob computes the encrypted distance $Enc(d(q, X_i), pk_C)$ for $i = 1, \ldots, N$; (2) Bob and Charlie jointly compute the encryption of each bit in the binary representation of $d(q, X_i)$; (3) Bob randomly sets $\epsilon' := \epsilon$ or $\epsilon' := 0$ with equal probability and computes $Enc(c_k^i, pk_C)$ for $i = 1, \ldots, N$ and $k = 1, \ldots, \lceil \log_2 n \rceil$, where $c_k^i = 0$ for any $k = 1, \ldots, \lceil \log_2 n \rceil$ indicates that $d(q, X_i) < \epsilon'$
3: Bob sends $Enc(c_k^i, pk_C)$ for $i = 1, \ldots, N$ and $k = 1, \ldots, \lceil \log_2 n \rceil$ to Charlie in random order.
4: Charlie decrypts them, computes $a := \prod_{i,k} c_k^i$ and sends back to Bob.
5: If $\epsilon' = 0$ and $a = 0$, or $\epsilon' = \epsilon$ and $a \neq 0$, Bob immediately rejects Alice and exits the program.
6: $M := M - 1$ and return to step 2.3 unless $M = 0$
7: Bob confirms Alice's membership

3. SECURE SIMILARITY SEARCH (SSS)

This section presents the implementation details of SSS, the key step in Algorithm 1. There are three sub-steps in SSS: Hamming distance computation, bit extraction, and threshold comparison, all of which are implemented in encrypted domain. In all the three sub-steps, both addition and multiplication are required in encrypted domain which can be satisfied by Paillier cryptosystem utilizing its key homomorphic prop-

erties [6]:

$$Enc(m_1, pk) \cdot Enc(m_2, pk) = Enc(m_1 + m_2, pk) \quad (1)$$

$$Enc(m, pk)^a = Enc(am, pk) \quad (2)$$

Also note that Paillier cryptosystem is one of the most commonly used homomorphic encryptions and provides semantic security, i.e., encrypting the same plaintext twice will result in two different ciphertexts with high probability.

3.1. Hamming Distance

The modified Hamming distance used for iris recognition in [5] is as follows:

$$d(x, y) = \frac{\| (x \otimes y) \cap mask_x \cap mask_y \|}{\| mask_x \cap mask_y \|} \quad (3)$$

In (3), $\otimes$ denotes XOR, $\cap$ AND, and $\| \cdot \|$ the norm of the binary vector. x and y are the binary vectors that represent the iris codes. $mask_x$ and $mask_y$ are the corresponding mask binary vectors that mask the unusable portion of the irises due to occlusion by eyelids and eyelash, specular refections, boundary artifacts of lenses, or poor signal-to-noise ratio. We assume that the mask vectors do not disclose identity information and thus Alice can send her mask vector in plaintext to Bob. As the division in Equation (3) may introduce floating point number, we focus on the following distance and roll the denominator into the similarity threshold during the later stage of comparison.

$$d(x, y) = \| (x \otimes y) \cap mask_x \cap mask_y \| \quad (4)$$

Algorithm 2 computes the hamming distance (4) between q and X_i for $i = 1, \ldots, N$. It is clearly secure as all the operations are done in the encrypted domain.

Algorithm 2 Hamming Distance

Require: Bob: $X_i, i \in \{1, 2, \ldots, N\}$, $Enc(q_k, pk_C)$, $Enc(NOT(q_k), pk_C), k \in \{1, 2, \ldots, n\}$

Ensure: Bob computes $Enc(d(q, X_i), pk_C)$ for $i = 1, \ldots, N$

For $i = 1, \ldots, N$, repeat the following two steps:

1: Compute $Enc(q_k \otimes X_k^i, pk_C)$ for $k = 1, \ldots, n$ using the following observation: $Enc(q_k \otimes X_k^i, pk_C) = Enc(q_k, pk_C)$ if $X_k^i = 0$ and $Enc(NOT(q_k), pk_C)$ otherwise

2: Compute $Enc(d(q, X_i), pk_C) = Enc\left(\sum_{k:[mask_q \cap mask_{X_i}]_k = 1} q_k \otimes X_k^i, pk_C\right) = \prod_{k:[mask_q \cap mask_{X_i}]_k = 1} Enc(q_k \otimes X_k^i, pk_C)$

3.2. Bit Extraction

The second step of SSS is to extract each bit from $d(q, X_i)$ for $i = 1, \ldots, N$ to be used in the threshold comparison. Even though q and X_i are n-dimensional binary vectors, the length of their hamming distance $d(q, X_i)$ is at most $\lceil \log_2 n \rceil$. As bit extraction cannot be expressed in terms of summation and multiplication, Bob requires the assistance of Charlie to carry out the task. The idea is to have Bob randomly perturb the data before sending them to Charlie for decryption – it is possible for Bob to compute $Enc(d(q, X_i) + r, pk_C)$ with a random number r in the encrypted domain. The plaintext $d(q, X_i) + r$, however, does not leak any information about $d(q, X_i)$ to Charlie. Charlie then performs the bit extraction, encrypts it and sends it back to Bob. The trick is for Bob to undo the perturbation. This can be done by starting the operation at the least significant bit (LSB) of $d(q, X_i)$. The recovery can be done by XORing $[d(q, X_i) + r]_1$ and r_1. We then zero out the LSB of $d(q, X_i)$ before moving onto the next round to recover the second LSB. As the LSB is zero, it has no effect to the second LSB and we can recover all the bits by repeating this procedure as described in Algorithm 3. The algorithm is secure because Charlie cannot gain new information as a new random number is used in each round, and Bob cannot gain new information as he only handles encrypted data.

Algorithm 3 Bit Extraction of $d(q, X_i)$

Require: Bob: $Enc(d(q, X_i), pk_C)$ and random source; Charlie: $Dec(\cdot, sk_C)$

Ensure: Bob computes $Enc([d(q, X_i)]_k, pk_C)$ for $k = 1, \ldots, \lceil \log_2 n \rceil$

1: Start with the least significant bit by setting $k := 1$ and $Enc(d, pk_C) := Enc(d(q, X_i), pk_C)$

2: Bob generates a random number r, computes $Enc(d + r, pk_C)$ and sends it to Charlie.

3: Charlie decrypts $Enc(d + r, pk_C)$, re-encrypts the k-th bit $Enc([d + r]_k, pk_C)$ and sends back to Bob.

4: Bob recovers $Enc(d_k, pk_C) = Enc([d(q, X_i)]_k, pk_C)$ by noting that it equals to $Enc([d + r]_k, pk_C)$ if $r_k = 0$ and $Enc(NOT([d + r]_k), pk_C)$ otherwise.

5: Bob zeros out the k^{th} bit of d by performing $Enc(d, pk_C) := Enc(d - d_k \cdot 2^{k-1}, pk_C) = Enc(d, pk_C) \cdot Enc(d_k, pk_C)^{-2^{k-1}}$

6: Set $k := k + 1$ and return to Step 2 unless $k = \lceil \log_2 n \rceil$

3.3. Threshold Comparison

In the final step of SSS, we need to determine if

$$d(q, X_i) < \epsilon' \cdot \| mask_q \cap mask_{X_i} \| \quad (5)$$

where $d(q, X_i)$ is the modified Hamming distance defined in (4). Based on the encrypted bit representation of $d(q, X_i)$ obtained in Section 3.2 and the actual bit representation $\epsilon_k^i := [\epsilon' \cdot \| mask_q \cap mask_{X_i} \|]_k$, we adopt the secure comparison protocol proposed by Damgard, Geisler and Kroigard in [8], which results in the computation of $Enc(c_k^i; pk_C)$ where $c_k^i = 0$ for some $k \in \{1, \ldots, \lceil \log_2 n \rceil\}$ if (5) is satisfied. The full algorithm is shown in Algorithm 4. We refer the reader to [8] for detailed explanation and proofs.

Algorithm 4 Secure comparison

Require: Bob: $Enc([d(q,X_i)]_k, pk_C), \epsilon_k^i$ for $k = 1, \ldots, \lceil \log_2 n \rceil$

Ensure: Bob computes $Enc(c_k^i; pk_C)$ such that $c_k^i = 0$ for some $k \in \{1, \ldots, \lceil \log_2 n \rceil\}$ if (5) is satisfied.

1: For $k \in \{1, \ldots, \lceil \log_2 n \rceil\}$, compute $Enc(w_k^i, pk_C) := Enc([d(q,X_i)]_k \otimes \epsilon_k^i, pk_C)$ by using the fact that $Enc(w_k^i, pk_C) = Enc([d(q,X_i)]_k, pk_C)$ if $\epsilon_k^i = 0$ and $Enc(NOT\left([d(q,X_i)]_k\right), pk_C)$ otherwise.

2: For $k \in \{1, \ldots, \lceil \log_2 n \rceil\}$, compute $Enc(c_k^i, pk_c) := Enc\left([d(q,X_i)]_k - \epsilon_k^i + 1 + \sum_{j=k+1}^{\lceil \log n \rceil} w_j^i, pk_C\right)$. This can be done in the encrypted domain as it involves only addition and subtraction.

4. EXPERIMENT

To evaluate the performance of the proposed system, we use the CASIA Iris database collected by the Chinese Academy of Sciences Institute of Automation (CASIA) [9]. Based on the Matlab feature extraction code from [10], we obtain the databases of both the iris codes and the mask codes. Each iris code is $n = 9600$ bit long. The gallery contains codes $N = 100$ users and the similarity threshold ϵ is set to 0.35. For each probe signal, we run the verification $M = 20$ times resulting in a probability of 10^{-6} in admitting an imposter. The key length of the public key cipher is set to be 1024 bit. Each 9600-bit iris code is encrypted bit by bit with the corresponding 2048-bit ciphertext. The networking time is assumed to be negligible compared to the computation time. It takes 27.1 minutes on average to perform the entire computation procedure for a single query on a linux machine with AMD Athlon 64, 2.4 GHz and 2GB memory. The time measurement for each of the key step in our algorithm for a single round of computation is provided in Table 1. The most time consuming process is the bit extraction since it includes multiple rounds of decryption and encryption with a new random number for each round in Algorithm 3. While the matching process is too slow to be used in real applications, our initial focus is the security of the system. We are currently investigating the optimization of various components of the system.

Table 1. Time measurement for one single round

Processing Steps	Time (second)
1. Alice encrypts q and $NOT(q)$	289.922
2. Secure Similarity Search	175.718
2.1 Hamming Distance	9.528
2.2 Bit Extraction	124.001
2.3 Threshold Comparison	42.189
3. Decryption of c_k^i	17.946

5. CONCLUSION

In this paper, we propose an anonymous biometric access control system that can anonymously authenticate the membership status of a user via his or her biometric signals. We believe that such a system should find widespread applications in e-commerce as users continue to demand stronger measure of privacy protection. There are a number of areas we are currently exploring to improve the performance of the system. For instance, by exploiting the fact that all the records in the gallery are compared with the same probe, we are studying how to pack multiple bits into the same ciphertext for parallel manipulation. Additionally, we plan to incorporate the improved Paillier homomorphic encryption in [8] which uses a smaller finite field for encrypted domain arithmetic for faster performance.

6. REFERENCES

[1] E. N. Newton, L. Sweeney, and B. Main, "Preserving privacy by de-identifying face images," *IEEE transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 232–243, February 2005.

[2] C. Diaz, *Anonymity and Privacy in Electronic Services*, Ph.D. thesis, Katholieke Universiteit Leuven, 2005.

[3] N.K. Ratha, J.H. Connell, and R.M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614–634, 2001.

[4] S. Hoque, M. Fairhurst, G. Howells, and F. Deravi, "Feasibility of generating biometric encryption keys," *Electronics Letters*, vol. 41, no. 6, pp. 309–311, 2005.

[5] John Daugman, "How iris recognitionworks," *IEEE Trans*, CSVT, 14(1), 21-30.

[6] P. Pailler, "Public-key cryptosystems based on composite degree residuosity classes," *Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 99)*, vol. vol. 1592, pp. 223–238, May 1999.

[7] J. Katz and Y. Lindell, *Introduction To Modern Cryptography*, Chapman and Hall, 2008.

[8] I. Damgard, M. Geisler, and M. Kroigard, "Homomorphic encryption and secure comparison," *International Journal of Applied Cryptography*, vol. 1, no. 1, pp. 22–31, 2008.

[9] T. Tan and Z. Sun, "Casia-irisv3," http://www.cbsr.ia.ac.cn/IrisDatabase.htm, 2005.

[10] Libor Masek and Peter Kovesi, "Matlab source code for a biometric identification system based on iris patterns," *The School of Computer Science and Software Engineering, The University of Western Australia.*, 2003.