International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **11**, November **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 8001–8018

PRIVACY-PRESERVING AUTHENTICATION USING FINGERPRINT

Quan Feng[1], Fei Su[2,3] and Anni Cai[2]

[1]Engineering College
Gansu Agricultural University
No. 1, Yingmen village, Anning Dist., Lanzhou 730070, P. R. China
fquan@sina.com

[2]School of Information and Telecommunication Engineering
[3]Beijing Key Laboratory of Network System and Network Culture
Beijing University of Posts and Telecommunications
P.O. Box 113, No. 10, Xitucheng Rd., Beijing 100876, P. R. China
{ sufei; annicai }@bupt.edu.cn

Received July 2011; revised December 2011

Abstract. *Biometrics provides us with an effective tool for authentication and is becoming a strong competitor in current authentication mechanisms. However, the concerns about biometrics regarding template security, privacy and revocability impede its further application to network-based scenarios. We propose a novel fingerprint minutiae-based scheme to achieve reliable mutual authentication over insecure channels. In the proposed scheme, a private template is generated which is completely random and independent of the minutiae, thus providing template protection, privacy preservation and revocability. A private matching scheme is embedded in an authentication protocol to ensure that both a server and a user can verify each other by matching the template against the query minutiae without revealing their respective inputs. We analyze the security of the proposed scheme under various attack scenarios. The experimental results on FVC2002-DB2 show that the scheme's verification accuracy is acceptable.*

Keywords: Fingerprint, Private matching, Privacy protection, Remote authentication

1. **Introduction.** In many Internet-based applications, remote authentication that establishes the identity of an entity under scrutiny is the first and the most critical link in the security chain. Reliable and secure authentication is thus of great importance. Traditional security models for identity verification are based on passwords and tokens. However, passwords are relatively weak because they are liable to be guessed, shared or even stolen, and if a token is lost, it is likely that its finder will log on the system [1]. Biometrics [2] employs unique physical characteristics, e.g., fingerprints, irises, faces, hand geometry and hand-written signatures, as a testimony to verify an identity, thus setting up direct and strong links between physical persons and their identities that is difficult to guess or forge. Therefore, biometric authentication gains obvious advantages over the traditional security methods. At the current time, biometrics is becoming more and more popular for remote authentication [2] and providing secure systems.

However, when biometrics is applied to the Internet for remote authentication, it faces great challenges in privacy and security. Firstly, storing biometric templates in plain text has raised many concerns of individual privacy – people are likely to worry that their biometric will be used in unreasonable and unaccountable ways, such as blatant misuse or "function creep" [2]. Moreover, since biometric data cannot be updated or canceled, once a person's biometric is compromised, it can never be used again. Secondly, the biometric data necessary for matching may have to pass through insecure networks where

they could be easily copied or stolen; this would threaten the security of the biometric data and the system. Thirdly, a server checks a user's biometric information to validate his identity. However, from the user's viewpoint, a remote server for authentication is also untrustworthy in an open network environment.

One of the solutions to protecting the privacy of a user's biometric data is to carry out biometric verification locally [3-8]. In these schemes, a user's biometric template is stored on a smart card and he shares a key with the server. During the login phase, the user inserts the card in a client's card reader, the client reads out the template, captures his biometric sample, and then matches the sample against the template. If the user passes the biometric verification, the smart card performs a cryptographic authentication protocol with the server using the shared key to prove the user's identity. Since there are no biometric data that need to be passed through the network, the user's privacy is naturally preserved. In this model, however, the local biometric verification and remote identity verification are completely decoupled. The server lacks a means to check the correctness of the user's biometric. These approaches leave chances open for impersonation attacks on the client, not addressing the problem of template protection.

Biometric template protection technique aims at maintaining a biometric template's privacy, addressing its revocation, and (re)generating keys from biometric data [8,9]. Some famous schemes of template protection include fuzzy commitment [10], fuzzy vaults [11] and fuzzy extractors/secure sketches [12,13]. Some remote authentication protocols have been proposed based on them [14-17]. Since biometric data are only approximately stable, to address the matching of two biometric samples secretly the error-correction concept is widely used in these schemes. However, this compels a strong condition on biometrics, which means some biometric samples matched with the templates in the traditional way may fail to match, thus degrading accuracy performance.

Recently, much work has focused on addressing biometric privacy by using the secure multi-party computation technique. The basic tools are homomorphic encryption and garbled circuits. According to different understandings about privacy, different strategies were taken.

Researchers [18,19] have investigated identity privacy and transaction anonymity; the former refers to the private relationship between a user's personalized name and corresponding biometric template, and the latter refers to a database that knows nothing about which user is authenticating himself to the service provider (server). In this model, users' biometric templates in plain text or in encrypted form are stored in a trusted database that is totally independent from service providers. For biometric verification, a user encrypts his biometric sample using a homomorphic encryption scheme and sends this to the server, which retrieves his index to be used in a Private Information Retrieval protocol between the server and the database. Finally, a decision is made after decryption or in the encryption domain by employing the homomorphic properties. However, the computation and communication complexities of this model are high.

Upmanyu et al. [20] considered template protection. In their work, the server does not store biometric template but stores the parameters of a user's personal classifiers, such as support vector machines or neural networks, which is trained in the plain feature space. The parameters are encrypted by the user's private key. Thus, biometric authentication is viewed as solving a two-class classification problem for each user separately. In the authentication phase, the server evaluates the user's biometric sample employing the classifier in the encryption domain. The method involves both multiplication and addition operations, requiring the usage of a doubly homomorphic encryption scheme (both additive and multiplicative homomorphic). Due to the lack of such a practical homomorphic encryption scheme, the protocol employed a multiplicative homomorphic encryption, and

simulated addition using a "clever" randomization scheme over one round of interaction between the server and the client. However, this may lead to some attacks. Though they took some remedial measures, some discussions about these measures are obscure and unconvincing.

The above methods do not provide mutual authentication mechanisms and thus cannot thwart the server's impersonation.

Computation of homomorphic encryption is a little costly. To put privacy-preserving biometric authentication into practice, more researchers just consider the privacy of transaction. In this model [21-25] a client holds a biometric X and a server holds a database D. The goal of these works is to learn whether X appears in D without revealing any information about X to the server, and without exposing anything about D to the client. These protocols include two phases: a distance computation phase in which the distances between X and the records in D are privately computed, and a matching phase in which one (or more) records matching X are picked out. Different secure computation methods were adopted in such work, [21,23,24] only employed homomorphic encryption while [22,25,26] used both homomorphic encryption and garbled circuits. Such protocols work in semi-honest security models, and template protection in the database is not considered.

Among all biometric characteristics, the fingerprint has one of the highest levels of reliability [27]. The schemes in [14-24,26] only work with ordered and length-fixed biometric features. However, some features, such as fingerprint minutiae, are inherently unordered, flexible-length and non-binary sets. Though fingerprints can be described as length-fixed features, e.g., FingerCodes [28], minutiae-based matching is more robust to distortions and frequently encountered in practical applications [27,29]. Furthermore, a minutiae template is significantly smaller than other template types on the average. A small template will help reduce the computation and communication loads in a network-based authentication system.

In this paper, we consider two aspects of privacy protection: 1) template protection and 2) transaction privacy. To reach the goals of privacy protection and security, we adopt two approaches. The first is different from existing biometric systems – a template for matching is completely independent of fingerprint minutiae, which is comprised of random numbers. The second allows the user and the server to establish identity of the other party through a mutual authentication protocol based on Private Matching Protocol (PM, or Private Set Intersection, PSI). All concerns about privacy, security and revocability can be addressed in the proposed scheme:

1) A template called reference set stored in the server has nothing to do with the fingerprint minutiae. Anyone cannot deduce the origin information of fingerprint. Thus, the privacy of a user's fingerprint is preserved. And since the reference set is generated randomly, one can replace the compromised template easily, thus providing ability of template revocability.
2) Only a user with the right smart card and right fingerprint is permitted to log into the server. This two-factor strategy makes the authentication more reliable.
3) Authentication is performed by matching fresh minutiae against the reference set through PM. The carefully designed authentication protocols based on PM are capable of protecting the transaction privacy of both the user and the server, thwarting various network attacks.
4) Both the server and the user are allowed to perform the minutiae-matching to verify each other. This feature makes our authentication protocol truly biometric-based and mutual, addressing concerns about trust between the user and the server.

The rest of this paper is organized as follows. Section 2 introduces the homomorphic encryption and PM protocol. Section 3 gives details of template generation. The remote mutual authentication scheme is described in Section 4. The security analysis of the proposed scheme is given in Section 5. The experimental results of the performance are presented in Section 6. Section 7 concludes the paper.

2. **Preliminaries.**

2.1. **Additively homomorphic encryption and evaluation on encrypted polynomial.**

Homomorphic encryption is a form of encryption where a specific algebraic operation performed on the plaintext is equivalent to another (possibly different) algebraic operation performed on the ciphertext. Let $E_K(\cdot)$ denote the encryption function with the public key K; an additively homomorphic encryption has two interesting properties – preserving the group homomorphism of addition and allowing multiplication by a constant without knowledge of the private key:

1) Given $E_K(a)$ and $E_K(b)$, one can efficiently compute the encryption of $a+b$, denoted as $E_K(a+b) := E_K(a) +_h E_K(b)$.
2) Given $E_K(a)$ and a constant c, one can efficiently compute $E_K(c \cdot a)$, denoted as $E_K(a \cdot c) := E_K(a) \times_h c$.

Any semantically-secure encryption can be used as a building block in our authentication protocols, e.g., Paillier [30] and Benaloh [31]. In Paillier's system, operation $+_h$ represents a multiplication and $\times_h$ represents an exponentiation. Thanks to the above two properties, given the encryptions of the coefficients $\{a_n, \ldots, a_1, a_0\}$ of a polynomial f whose degree is n, and the knowledge of a plaintext y, it is possible to compute an encryption of $f(y)$ without use of the private key. We denote the encryption of f by $E_K(f)$ and define it to be the list of encryptions of its coefficients: $\{E_K(a_n), \ldots, E_K(a_1), E_K(a_0)\}$. Then we can evaluate $E_K(f(y))$ as follows:

$$\begin{aligned} E_K(f(y)) &= E_K\left(\sum_{i=0}^{n} a_i \cdot y^i\right) = E_K(a_n y^n) +_h \cdots +_h E_K(a_1 y) +_h E_K(a_0) \\ &= E_K(a_n) \times_h y^n +_h \cdots +_h E_K(a_1) \times_h y +_h E_K(a_0) \end{aligned}$$

2.2. **Private matching protocol.**

Private Matching protocol (PM) or Private Set Intersection protocol [32-34] aims at addressing several two-party set-intersection problems and enables the two parties holding a set of inputs each to jointly calculate the intersection of their inputs without leaking any additional information to each other. Traditionally, PM is designed for those scenarios where private data must be shared between two mutually suspicious entities, e.g., Private Information Retrieval (PIR). Freedman et al. [32] constructed efficient PM protocols by means of oblivious polynomial evaluation and additively homomorphic encryption. The computational complexities of the server and the client are only $O(mn)$ and $O(n)$ respectively. In this paper, we utilize Freedman's scheme to construct a remote fingerprint authentication system.

3. **Template Generation.**

In our scheme, minutiae are used to verify a user's identity. Minutiae are the most prominent ridge characteristics on a fingertip, characterized by either endings or bifurcations of the ridges. A minutia is usually represented as (x, y, θ), where (x, y) is the location of the minutia and θ is the orientation of its associated ridge. Commonly, minutia sets that originate from the same finger do not contain the exactly same minutiae due to errors in capturing, image processing and so on. The matching of two minutia sets is usually posed as a point-pattern matching problem. In traditional minutiae-based verification systems, the similarity between the query minutiae and the

template is proportional to the number of matched minutiae pairs, and a match of two fingerprints is usually claimed when this number reaches or exceeds a threshold [27].

In network applications, it is desirable that both a server and a user can verify each other through a fingerprint-matching. Obviously, a template stored in the server must be kept in encrypted forms or significant randomness to prevent the server from learning any information about the user's fingerprint. We choose the latter and set the following security requirements on constructing the new template:

1) The elements of the template should be random and uniformly distributed. Thus, it is impossible for malicious servers to guess the user's biometric information.
2) Distinct templates generated from the same minutia should be mutually independent. This feature endows the template the capability of revocability.
3) Since the range of (x, y, θ) of a minutia lies on a relatively small domain, it is vulnerable to an exhaustive search attack. Accordingly, to resist this attack, the values of elements in the template should be distributed in a large domain.

To meet the above requirements 1 and 2, we simply generate some random numbers to form a template. Because the template is completely independent of the minutiae template and totally random, we prefer to call it a *reference set* in this paper. Thus, the set possesses perfect properties of privacy protection and revocability. In authentication, we match directly the query minutiae against the reference set. To address the problem of matching two sets without any relation, we introduce a transform polynomial, which is derived jointly from the reference set and the user's minutiae template. Figure 1 shows the block diagram of template generation.

Before template generation, minutiae must be quantized to account for the slight variations in minutiae data. For a minutia (x, y, θ), assume $x \in [0, X]$, $y \in [0, Y]$ and $\theta \in [0, 360)$. We divide the attribute space into N bins (subspaces), namely, $N = \left\lfloor \frac{360 \cdot X \cdot Y}{q_\theta \cdot q_x \cdot q_y} \right\rfloor$, where q_θ, q_x and q_y are quantization parameters in θ, x and y respectively. These parameters can be denoted by a triple $Q = \{q_\theta, q_x, q_y\}$. Each bin is then be indexed by an integer. Let $\mathcal{I}' = \{1, 2, \ldots, N\}$ denote all the bins. If a minutia falls into the ith bin, it is labeled with i. Due to quantization, some distinct minutiae may fall into one bin, so they will be indexed by the same integer. To obviate this, we enlarge N. Suppose that N is represented by n_l bits. We add n_g bits ahead of the n_l bits to represent the multiple minutiae in one bin. For example, if $n_g = 3$, we can index eight minutiae within the same bin by distinct integers. Let $\mathcal{I}$ denote the enlarged set of index. After quantization, a

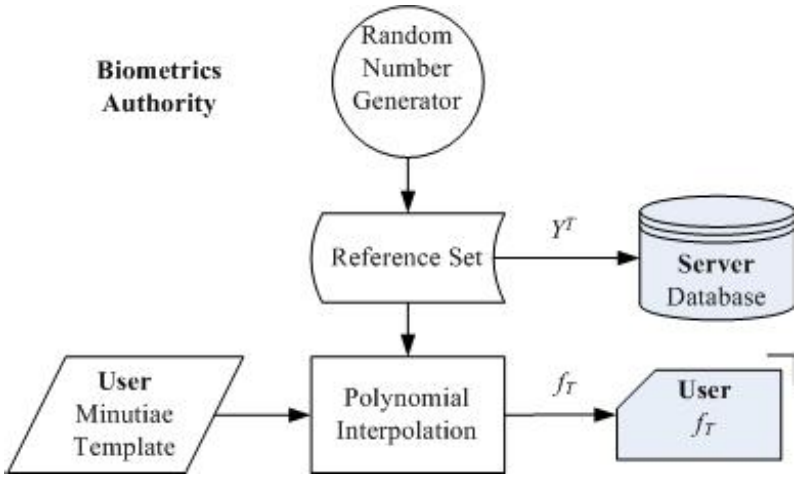

FIGURE 1. Block diagram of template generation

minutiae template is converted to a set of integers and denoted as $U^T = \{u_i^T | u_i^T \in \mathcal{I}\}_{i=1}^{N_T}$ where N_T is its cardinality.

Now, we detail the template generation as follows.

Public parameters: an integer k, a finite field $\mathcal{F}$, a hash function $H(.)$.

Algorithm PVTM

Input: U^T, an integer $ID \in \mathcal{F}$.

Output: a reference set Y^T and a transform function f_T.

1) If $N_T \geq k$, continue; otherwise, output {"*null*"} and exit.
2) Choose $y_1, y_2, \ldots, y_{N_T}$ independently and randomly with uniform distribution from $\mathcal{F}$.
3) Use Lagrange interpolation to construct a polynomial f_T of degree $N_T - 1$ from $\{(u_i^T, y_i)\}_{i=1}^{N_T}$:

$$f_T(x) = \sum_{i=1}^{N_T} y_i l_i(x), \text{ where } l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{N_T} \frac{x - u_i^T}{u_i^T - u_j^T}.$$

4) Compute $C = H(f_T(ID))$.
5) Output $\{f_T, Y^T = \{y_1, y_2, \ldots, y_{N_T}\}, C\}$.

Note that all polynomial operations in this paper take place in $\mathcal{F}$. To meet the aforementioned requirement 3, $|\mathcal{F}|$ should be much greater than $|\mathcal{I}|$. Y^T is the reference set. In order not to leak any information about the order of the elements in U^T, the elements of Y^T may be output in a pre-determined order, or in a random order. $C = H(f_T(ID))$, where $H()$ is a collision-resistant hash function, can be used as the supplementary testimony for a user's identity (the detail is given in Section 4.4). According to Proposition 5.1 in Section 5, the coefficients of the transform polynomial f_T are random and uniformly distributed over $\mathcal{F}$ without leaking any information of U^T. The integer k is a pre-defined threshold that controls the required closeness between query minutiae U^Q and the template U^T, that is, k is the required minimum number of elements in the intersection of two sets for declaring them matched. k also determines the security of the system against brute-force attack. The bigger k is, the more secure the system is. More security analysis of the system will be given in Section 5.

Let us make a sketch about our authentication scheme: Y^T is distributed to the server while f_T is sent to the user (stored in a smart card) in advance. On the client side, the user employs f_T to transform his indexed query minutiae U^Q into a private set Y^Q and send it to the server. The server validates the user's identity by matching Y^Q against Y^T, and then the user must do the same work to check if the server is desired. This process needs a reliable cryptographic protocol to guarantee the security and privacy. We will design such a protocol in next section. The pre-alignment between the query and the template poses a common challenge. Fortunately, several approaches [35-39] have been presented to address this problem. For the purposes of this paper we assume the query has been aligned with the template.

According to the construction of f_T and Y^T, we can see that if $u_i^Q \in U^Q$ belongs to the intersection of U^Q and U^T, $f(u_i^Q)$ certainly belongs to Y^T. If $u_i^Q \in U^Q$ does not belong to the intersection of U^Q and U^T. According to Lemma 5.1 in Section 5, $f(u_i^Q)$ is random and uniformly distributed over $\mathcal{F}$, so there exists a chance that the u_i^Q can be transformed into an element of Y^T. Obviously, this error probability is $1/|\mathcal{F}|$. If $|\mathcal{F}|$ is sufficiently large, the error probability can be negligible. For instance, the error probability approximates 5.42×10^{-20} for $|\mathcal{F}| = 2^{64}$. Hence, for a large $|\mathcal{F}|$, matching

Y^T against Y^Q is essentially equivalent to matching U^T against U^Q, and vice versa. As a result, we have $|Y^T \cap Y^Q| = |U^T \cap U^Q|$ and define that the server and the user verify each other if $|Y^T \cap Y^Q| \geq k$ (namely, $|U^T \cap U^Q| \geq k$).

4. Remote Authentication Protocol.

In this section, we propose a mutual remote authentication scheme in which matching is performed both at the server side and at the client side. The proposed scheme includes two phases: 1) enrollment, and 2) authentication.

4.1. Security models.

The proposed authentication system consists of four types of components:

1) A biometrics authority (BA), which is a trusted party in charge of preparing the public parameters required in the scheme and template generation.
2) A user who first registers his minutia template to BA, and later uses his query fingerprint to authenticate himself to the server. The user is in possession of a smart card, which stores authentication information issued by BA. The smart card performs the authentication protocol with the server.
3) A client equipped with a biometric module, a smart card reader, a communication terminal, etc., is capable of capturing the user's fingerprint, extracting, pre-aligning and quantizing the minutiae, and communicating with the server.
4) A server stores users' reference sets and deals with the users' identity verification.

Our focus is on security issues concerning the communication link between the user and the server during authentication, and the privacy of the verification data. We assume that the biometric module, the smart card reader and the terminal at the client are all tamper-proof devices, and any transmission between them is secure. We do not discuss the impact of using fake fingers to fool the system. The counter-measure is related to liveness detection [40], which is beyond the scope of this paper. We further assume that the server and the client in the system will not collude with each other. We expect the proposed authentication protocol to achieve the following security goals:

1) Correctness: The protocol should be correct, i.e., the server verifying the claimed identity only if the authentication request is from a user with the right biometric and the right smart card, while the user verifies the server only if the server returns the right verification data.
2) Privacy: While the protocol is being implemented, except for the matching elements, the two parties cannot get any additional information, and the information gathered by an adversary cannot help him to restore the user's original fingerprint template.

4.2. Enrollment.

BA must prepare the public parameters, such as a pre-determined k, $\mathcal{F}$, the quantization parameters Q, a hash function $H(\cdot)$ and a size parameter g, which is the upper bound of set size used in the proposed protocol, as well as the parameters for the additively homomorphic encryption schemes that are used in the protocol. In the enrollment phase, a user applies to BA for registering himself to server S_n, BA assigns him an identity number (ID) U_m, captures his fingerprint sample and extracts the minutiae template M^T. Then, BA quantizes M^T to U^T with Q and invokes algorithm Pvtm (U^T, S_n) and outputs $\{f_T, Y^T, C\}$. He sends $\{U_m, Y^T, C\}$ to the server S_n in a secure manner and the server stores these data in a safe database. The public parameters $\{\mathcal{F}, k, H(\cdot), Q, g\}$, the parameters for the additively homomorphic encryption scheme as well as private parameters f_T (its coefficients, actually), U_m and S_n are stored on a smart card that is issued to the user. BA must provide a public key certificate (PKC) and a private key for a user, which can be obtained from a certificate authority (CA). They are

also stored in the user's smart card. At the end of the enrollment, BA discards all the biometric data about the user, such as Y^T and f_T.

4.3. **Private matching protocol.** In order to achieve the privacy-preserving goals, Freedman's basic PM scheme [32], which is secure in semi-honest model [41] is employed to construct the proposed authentication protocol. The reasons are based on the following considerations. Firstly, there has not been a "really" efficient PM protocol that can prevent a party from misbehaving. Secondly, for Freedman's scheme, attacks initiated by a malicious party will only influence the protocol's correctness, rather than its privacy. And in our authentication protocol the data used for matching are in transformed versions (Y^Q and Y^T), not original fingerprint data. Thirdly, we will modify Freedman's basic PM scheme a bit to fit our authentication protocol and resist some malicious attacks.

Let denote a server S_n and a user U_m respectively. We now describe the modified PM algorithm used in the authentication protocol.

Public parameters: a finite field $\mathcal{F}$, a size parameter g, a pair of additively homomorphic encryption/decryption algorithm $E(\cdot)/D(\cdot)$.

Algorithm PRIVATE-MATCHING

Input: $Y = \{y_1, y_2, \ldots, y_{N_S}\}$, K_S and v are S_n's input set, public key and a random number respectively; $X = \{x_1, x_2, \ldots, x_{N_C}\}$ is U_m's input set.

Output: S_n gets the result of $G = X \cap Y$.

1) S_n carries out the following operations:
   (a) Compute a polynomial $f_C(y) = (y-y_1)(y-y_2)\ldots(y-y_{N_S}) = y^{N_S} + a_{N_C-1}y^{N_S-1} + \ldots + a_1 y + a_0$.
   (b) Compute $cf_i = E_{K_S}(a_i)$, $0 \le i \le N_{S-1}$.
   (c) Send $\{cf_0, \ldots, cf_{N_S-1}\}$ to U_m.
2) If N_S is bigger than g, U_m aborts the protocol; otherwise, perform the following operations:
   (a) Utilize the homomorphic properties to evaluate the encrypted polynomial at each $x_i \in X$:
   $$E_{K_S}(f_C(x_i)) = E_{K_S}(1) \times_h (x_i)^{N_C} +_h \cdots +_h E_{K_S}(a_1) \times_h x_i +_h E_{K_S}(a_0),$$
   $$1 \le i \le N_C.$$
   (b) Choose a random r_i from $\mathcal{F}$ each $x_i \in X$ and calculate $cq_i = E_{K_S}(r_i \cdot f_C(x_i) + x_i) = E_{K_S}(r_i \cdot f_C(x_i)) +_h E_{K_S}(x_i) = E_{K_S}(f_C(x_i)) \times_h r_i +_h E_{K_S}(x_i)$ again using the homomorphic properties, $1 \le i \le N_C$.
   (c) Calculate $cr_i = cq_i \oplus v$, $1 \le i \le N_C$.
3) U_m randomly permutes the N_C resulted ciphertexts, and sends them back to S_n.
4) If N_C is bigger than g, S_n aborts the protocol; otherwise, he computes $cq_i = cr_i \oplus v$, $1 \le i \le N_C$, and decrypts all these N_C ciphertexts to get a set $Y' = \{y'_1, \ldots, y'_{N_C}\}$.
5) S_n finds the intersection G between Y' and Y ($G = X \cap Y$).

In PRIVATE-MATCHING, the server prepares the encrypted polynomial, the user evaluates it and the server gets the intersection while in [32], the thing is turned over. The proof of correctness and privacy of PRIVATE-MATCHING are the same as in [32]. In addition, we create a random v to provide fresh results returned to the server. g is a size parameter, whose role will be introduced in Section 5 in detail. The primary computation of the server is the encryption and decryption of N_S coefficients. Since the server's input set is fixed, the polynomial f_C and $E_{K_1}(f_C)$ can be pre-computed by the server and thus only the decryption with complexity of $O(N_S)$ is really needed during the implementation of the protocol. The primary computation load of the user is to compute $E_{K_1}(r_i \cdot f_C(x_i) + x_i)$ which contains $O(N_S N_C)$ exponentiations. As exponentiation dominates the computation

of the user, Freedman [32] proposed several effective methods to reduce the number of exponentiations, e.g., using Horner's rule, and using hashing for bucket allocation. The user's computational complexity can then come down to $O(N_S + N_C \ln\ln N_S)$ exponentiations. The main communication load of PRIVATE-MATCHING is determined by the total number of coefficients of the polynomial and its evaluations on X, namely, $O(N_S + N_C)$.

4.4. **Privacy-preserving authentication based on fingerprint.** Though PRIVATE-MATCHING is secure in semi-honest model our authentication protocol also shows the effectiveness against some malicious attacks, such as men-in-the-middle attacks (MITM), replay attacks and impersonation attacks. Figure 2 shows the block diagram of the proposed authentication protocol. Of course, if the server is trusted, the server's identity validation procedure can be ignored, and the proposed authentication protocols will be simplified remarkably. For simplicity, we do not discuss this case.

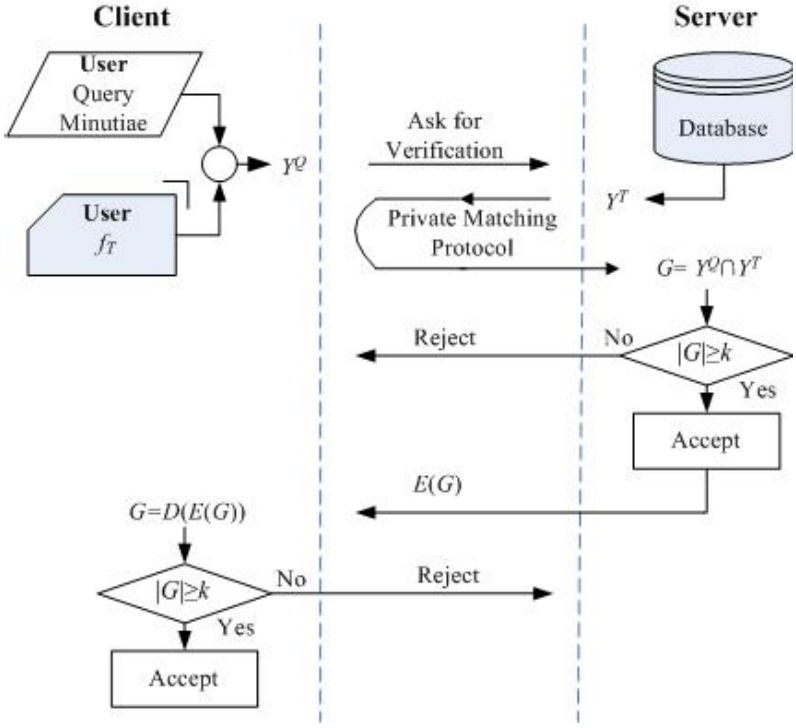

FIGURE 2. Block diagram of proposed authentication protocol

When a user U_m wants to authenticate himself to a remote server S_n through a client, he inserts his smart card into the card reader, and imprints his fingerprint on the finger scanner. The client extracts the query minutiae M^Q, pre-aligns the query with the template, and quantizes them with Q to obtain U^Q. U_m then computes $Y^Q = f_T(U^Q) = \{y_i^Q\}_{i=1}^{N_Q}$ where $y_i^Q = f_T(u_i^Q)$, and asks S_n for verification. Subsequently a mutual authentication protocol, CK-AUTHENTICATION, is implemented between S_n and U_m. The details of CK-AUTHENTICATION are described as follows. For simplicity, we assume that the two parties have exchanged their respective public key and the server has certified the user's PKC.

Public parameters: a threshold parameter k, a finite field $\mathcal{F}$, a hash function $H(\cdot)$, a pair of additively homomorphic encryption/decryption algorithms $E(\cdot)/D(\cdot)$

Protocol CK-AUTHENTICATION

Input: U_m's input is set Y^Q and public key K_U; S_n's input is set Y^T, C and public key K_S.

1) U_m sends his authentication request together with ID (U_m) to S_n.
2) S_n retrieves the corresponding $\{Y^T = \{y_1, y_2, \ldots, y_{N_T}\}, C\}$ from the safe database with respect to U_m, chooses a random number v from $\mathcal{F}$ and sends $ch = E_{K_U}(v)$ to U_m.
3) U_m calculates $v = D(ch)$ and $C_E = E_{K_S}(f_T(S_n) \oplus v)$, then returns C_E to S_n.
4) S_n computes $H(D(C_E) \oplus v)$, and checks if it is just equal to C, if no, reject U_m.
5) S_n and U_m implement Private-Matching (Y^T, K_S, v, Y^Q). If no exception happens, at the end of Private-Matching, S_n gets the matching result $G = Y^T \cap Y^Q$.
6) If $|G| \geq k$, S_n accepts U_m's identity; otherwise, reject U_m.
7) S_n randomly permutes G, compute $G' = \{g'_i = g_i \oplus v | g_i \in G\}$ and sends $E_{K_U}(G') = E_{K_U}(g'_1||g'_2||\ldots||g'_{|G|})$ to U_m.
8) U_m decrypts the received ciphertexts to get G', compute $G = \{g'_i \oplus v | g'_i \in G'\}$. Then, U_m checks if G is a subset of Y^Q and if $|G| \geq k$, if yes, U_m verifies S_n.
9) The two parties respectively compute $K = H(G||v) = H(g_1||g_2||\ldots||g_{|G|}||v)$ as the session key for the later use.

In CK-Authentication, the server compares Y^Q and Y^T, the transformed versions of U^Q and U^T, by means of PM. After verifying the user, the server returns G to the user. "||" in step 7 denotes concatenation of the bit strings. The user verifies the server also by checking G. Since at the end of the protocol, the two parties both learn their intersection G, the authentication protocol is actually a mutual PM. However, it is a conditioned mutual PM, since if and only if the condition of $|G| \geq k$ is satisfied, the server returns G to the user. The hash function $H()$ used in step 4 must be the same as that in Pvtm.

According to the statistics, the number of minutiae in a fingerprint ranges from 20 to 100 with high probability [40]. The computation spent on matching two sets is very little in CK-Authentication due to the small number of minutiae in one fingerprint. The primary computation and communication loads come from the execution of Private-Matching, which has been discussed in Section 4.3.

5. Security Analysis.

5.1. Privacy of minutiae template.

Now we analyze the privacy of the minutiae template U^T. The algorithm Pvtm outputs a reference set $Y^T = \{y_1, \ldots, y_{N_T}\}$ and a transform function $f_T = c_{N_T-1}u^{N_T-1} + \cdots + c_1 u + c_0$, where $y_1, \ldots, y_{N_T}$ are mutually independent and uniformly distributed over $\mathcal{F}$, and f_T is derived from U^T and Y^T. Our strategy for storage is that f_T is stored on the smart card held by the user, and Y^T is stored in the server. The generation method and the storing strategy guarantee that an adversary cannot learn anything from Y^T or f_T. To detail the analysis of security, we first introduce several propositions.

Lemma 5.1. [43]. *Let $X_1, \ldots, X_t$, X be mutually independent and uniformly distributed over a finite filed $\mathcal{F}$, and $a_1, \ldots, a_t \in \mathcal{F}$, $Y_{a_1,\ldots,a_t} := a_1X_1 + \ldots + a_tX_t + X$, then $Y_{a_1,\ldots,a_t}$ is uniformly distributed over $\mathcal{F}$, and the collection of all such $Y_{a_1,\ldots,a_t}$ is pairwise independent.*

Proposition 5.1. *Let polynomial f_T in* Pvtm *be $f_T(x) = c_{N_T-1}x^{N_T-1} + \cdots + c_1x + c_0$, then $c_{N_T-1}, \cdots, c_1, c_0$ are random variables uniformly distributed over $\mathcal{F}$.*

Proof: In Pvtm, f_T is generated from $\{(u_i^T, y_i)\}_{i=1}^{N_T}$ by Lagrange interpolation, hence,

$$f_T(x) = \sum_{i=1}^{N_T} y_i \prod_{\substack{j=1 \\ j \neq i}}^{N_T} \frac{x - u_i^T}{u_i^T - u_j^T}$$

$$= \sum_{i=1}^{N_T} y_i \sum_{k=0}^{N_T-1} p_k^i(u_1^T, \ldots, u_{N_T}^T)x^k$$
$$= \sum_{k=0}^{N_T-1} \sum_{i=1}^{N_T} p_k^i(u_1^T, \ldots, u_{N_T}^T)y_i x^k$$

Then, $c_k = \sum_{i=1}^{N_T} p_k^i(u_1^T, \ldots, u_{N_T}^T)y_i$, $0 \le k \le N_T - 1$.

For each c_k, we choose a $p_k^i \neq 0$ among p_k^is where $i = 1, \ldots, N_T$. Without loss of generality, we assume that $p_k^{N_T} \neq 0$. Then, $c_k = p_k^{N_T}\left(y_{N_T} + \frac{p_k^{N_T-1}}{p_k^{N_T}} y_{N_T-1} + \cdots + \frac{p_k^1}{p_k^{N_T}} y_1\right)$. Note that we calculate $\frac{p_k^i}{p_k^{N_T}}$ over $\mathcal{F}$, and $y_1, \ldots, y_{N_T}$ are uniformly distributed over $\mathcal{F}$. Therefore, according to Lemma 5.1, c_k is also uniformly distributed over $\mathcal{F}$, where $0 \le k \le N_T - 1$.

Proposition 5.2. *Given an ensemble of reference sets $\{Y_1^T, Y_2^T, \ldots, Y_n^T\}$ of the same U^T, resorting to it, the probability of guessing U^T is just equal to that of directly guessing U^T, namely,* $\Pr(U^T|Y_1^T, Y_2^T, \ldots, Y_n^T) = \Pr(U^T)$.

Proof: $\Pr(U^T|Y_1^T, Y_2^T, \ldots, Y_n^T) = \Pr(U^T, Y_1^T, Y_2^T, \ldots, Y_n^T)/\Pr(Y_1^T, Y_2^T, \ldots, Y_n^T)$. Since Y_i^Ts are chosen independently and randomly from $\mathcal{F}$ and unrelated to U^T, $\Pr(U^T, Y_1^T, Y_2^T, \ldots, Y_n^T) = \Pr(U^T)\Pr(Y_1^T, Y_2^T, \ldots, Y_n^T)$. Hence, the result is obvious.

Proposition 5.3. *Given an ensemble of transform polynomials $\{f_T^1, f_T^2, \ldots, f_T^n\}$ of the same U^T but without the corresponding $\{Y_1^T, Y_2^T, \ldots, Y_n^T\}$, it is impossible to solve U^T from the ensemble.*

Proof: We denote f_T^j with its coefficients, namely, $\{c_{N_T-1}^j, \cdots, c_1^j, c_0^j\}$, $0 \le j \le n$. According to the proof of Proposition 5.1, $c_k^j = \sum_{i=1}^{N_T} p_k^i(u_1^T, \ldots, u_{N_T}^T)y_i^j = f_k(u_1^T, \ldots, u_{N_T}^T, y_1^j, \ldots, y_{N_T}^j)$, where $\{y_1^j, \ldots, y_{N_T}^j\}$ is the jth reference set. Then, for f_T^j, we have the following equations:

$$\begin{cases} c_{N_T-1}^j = f_{N_T-1}\left(u_1^T, \ldots, u_{N_T}^T, y_1^j, \ldots, y_{N_T}^j\right) \\ \quad\quad \cdots \\ c_0^j = f_0\left(u_1^T, \ldots, u_{N_T}^T, y_1^j, \ldots, y_{N_T}^j\right) \end{cases}$$

Obviously, the number of the unknown quantities in the above equations is $2N_T$ while the number of the known quantities is only N_T. Therefore, $\{u_1^T, \ldots, u_{N_T}^T\}$ cannot be solved. If one more polynomial, f_T^{j+1}, is introduced, it will increase N_T known quantities, namely, $c_{N_T-1}^{j+1}, \cdots, c_1^{j+1}, c_0^{j+1}$. In the meantime, it will also increase N_T unknowns, namely, $y_1^{j+1}, \ldots, y_{N_T}^{j+1}$. Since the number of the unknown is always N_T more than that of the known, no matter how many f_Ts are combined, the equations cannot be solved.

Proposition 5.4. *Given an ensemble of $\{Y_1^T, Y_2^T, \ldots, Y_n^T\}$ of the same U^T but without U^T, it is impossible to solve $\{f_T^1, f_T^2, \ldots, f_T^n\}$.*

(The proof is similar to Proposition 5.3. The result is obvious, since the number of the unknown is $2N_T$ while the number of the known is N_T, the equations cannot be solved.)

Now, we consider the following attack scenarios:

1) A malicious server attempts to directly guess U^T or f_T using Y^T. Since Y^T is managed by a server, a malicious server will do so without hesitancy. However, according to Proposition 5.2, he cannot get any information about U^T from Y^T. Due to f_T

generated jointly from Y^T and U^T, if the server can guess the right f_T, he will recover U^T by means of polynomial factoring in $\mathcal{I}$ the domain of U^T) as below:

$$f_T(u) - y_i^T = c_{N_T-1}u^{N_T-1} + \cdots + c_1 u + c_0 - y_i^T = 0, \quad 1 \le i \le N_T$$

However, it is hard for the server to guess the correct f_T using Y^T according to Proposition 5.4.

2) A malicious server tries to get information about U^T or f_T in the process of performing CK-Authentication. In the protocol, the data sent to the server are in their transformed versions, and the protocol ensures that the server obtains no more information than the intersection between the reference set and the transformed query set. Thus, the server still cannot recover f_T or U^T.
3) An attacker attempts to recover U^T if he cracks a smart card and gets f_T. Though f_T is created by interpolation from $\{(u_i^T, y_i)\}_{i=1}^{N_T}$, the attacker can learn nothing about U^T from f_T according to Proposition 5.3.
4) A user may register in several servers, so each server maintains a reference set of the user. If multiple servers want to collude and guess U^T or f_T, according to Proposition 5.2, no additional information about U^T can be derived from their Y^Ts. It is also impossible to recover f_T according to Proposition 5.4.
5) When a user registers in several servers, he receives multiple smart cards. If an attacker gathers these smart cards and tries to restore the original fingerprint no matter how many f_Ts he gets, according to Proposition 5.3, he does not recover U^T.

5.2. Security of the proposed authentication protocol.

Proposition 5.5. *Let A and B be two multi-sets whose elements are independently and randomly picked out from a set $\mathcal{U}$, and $n = |A|$, $m = |B|$, then the probability that B at least shares k common elements with A is $\frac{C(n,k)}{|\mathcal{U}|^k}$.*

Proof: Since the elements of A and B are chosen independently and randomly, the probability of a value chosen from $\mathcal{U}$ is equal to $1/|\mathcal{U}|$. The numbers of possible patterns of A and B are $|\mathcal{U}|^n$ and $|\mathcal{U}|^m$ respectively. For a specific pattern of A, if B is expected to share at least k elements with A, B can be constructed as follows: 1) Choose randomly k elements of A and put them into B; 2) Choose independently and randomly $m-k$ elements from $\mathcal{U}$ and add them into B. The number of possible combinations of 1) is $C(n,k)$ while the number of 2) is $|\mathcal{U}|^{m-k}$. Hence, the probability p of B sharing at least k elements with A is:

$$p = \frac{|\mathcal{U}|^n \cdot C(n,k) \cdot |\mathcal{U}|^{m-k}}{|\mathcal{U}|^n \cdot |\mathcal{U}|^m} = \frac{C(n,k)}{|\mathcal{U}|^k}.$$

The security of Freedman's PM had been proven in [32]. Now, we discuss some attack cases against the proposed authentication protocol

1) **Men-in-the-middle attack.** In CK-Authentication, an attacker can intercept the initial public key exchange and substitute his forged key for the client's key sent to the server. If there is no countermeasure, when the server returns encryption of G (step 7 of CK-Authentication) with the forged key, the attacker learns G. He will use this information to successfully impersonate the legitimate user in the future. However, we employ PKC to defeat this attack. Under the framework of Public Key Infrastructure (PKI) [42], the user's public key can be certified by validating the PKCs, so a men-in-the-middle attack (MITM attack) is not possible.

2) **Replay attack.** We employ challenge-response mechanisms to defeat replay attacks. When the server wants to validate the user's identity, it initiates Private-Matching to learn the matching results from the user. The results are encrypted by the server's

public key. An attacker may record messages in the past and use them to launch a replay attack, because the key is long-term. To prevent such an attack, the server creates a random number v and encrypts it with the user's public key (step 2 of CK-Authentication). In Private-Matching, the user XORs his answers with v (step 2(c) of Private-Matching) to provide the required freshness. Similarly, in order to prevent the attacker from fooling the server with old C_E, the user XORs $f_T(S_n)$ with v (step 3 of CK-Authentication).

3) **Masquerade attacks of user.** If an impostor impersonates a legitimate user and asks for the server's authentication, he needs to create a query set Y, which shares at least k elements with Y^T. Assume he has no smart card, and independently and randomly picks out elements from $\mathcal{F}$ to construct Y. Since the elements of Y^T are also chosen independently and randomly from $\mathcal{F}$, according to Proposition 5.5, his chance of being successfully verified by the server is $\frac{C(N_T,k)}{|\mathcal{F}|^k}$. For $|\mathcal{F}| = 2^{64}$, $N_T = 40$ and $k = 12$, the probability is $\frac{C(40,12)}{|2^{64}|^{12}} = 5.6 \times 10^{-222}$. Hence, his chance of success is negligible.

If the impostor steals a user's smart card, there is no need for him to guess Y^T, but just U^T, i.e., he needs to match at least k elements of U^T. Since elements of U^T belong to $\mathcal{I}$ whose cardinality $|\mathcal{I}|$ is much smaller than $|\mathcal{F}|$ and these elements are distinct, guessing U^T is easier than guessing Y^T. If the elements of U^T are distributed uniformly over $\mathcal{I}$, the probability of success is $C(N_T, k)/C(|\mathcal{I}|, k)$. For $|\mathcal{I}| = 4000$, $N_T = 40$ and $k = 12$, the probability is 1.6×10^{-25}. It approaches the probability of guessing a password, the length of which is 16 characters, whose elements are arbitrarily taken among 26 letters and 10 digits (the latter probability is $1/36^{16} \approx 1.26 \times 10^{-25}$). However, it should be noted that minutiae are not distributed uniformly, thus the imposter's probability of success is a little higher than the above estimated value. Perhaps a simple way for the impostor to gain access is to provide his own fingerprint to the server when he gets the smart card. The non-zero FAR (false accept rate) of a practical biometric system leaves him little chance of passing the authentication. Of course, his chance is determined by the similarity between his U^Q and the legitimate user's U^T. The FAR of the proposed scheme will be tested in Section 6.

If Y^T stored in the server is leaked out to an imposter for some reasons, such as a hijacker's intrusion then, in this case, PM cannot prevent him from impersonating. However, we designed a countermeasure to preclude him from doing so, in which a supplementary testimony $C = H(f_T(ID))$ (step 4 in Pvtm) is generated in the enrollment phase. In the authentication phase, the imposter must calculate $f_T(S_n)$ (step 3 of CK-Authentication). If he is not in possession of the correct smart card, he does not know the correct f_T, and he surely cannot compute the correct value of $f_T(S_n)$. Thus the server can detect his impersonation (step 4 of CK-Authentication).

4) **Masquerade attacks of server.** Since a user also validates a server's identity by checking the intersection between Y^Q and Y^T, if a fake server without Y^T wants to impersonate the right one, he must guess at least k elements of Y^Q. Likewise, this probability is $\frac{C(N_Q,k)}{|\mathcal{F}|^k}$ according to Proposition 5.5, so his chance of success is also negligible.

In open networks, a user can register to several servers for different services. This may provide a chance for a malicious server S_i to impersonate the user's target server S_n, but our scheme can resist this attack. At the enrollment phase, BA creates $\{Y_i^T, f_i^T\}$ for S_i and $\{Y_n^T, f_n^T\}$ for S_n. If the user U_m asks S_n for authentication, S_i intercepts the request message, blocks S_n and sends U_m the encrypted polynomial $E(f_C)$ generated by Y_i^T. U_m calculates $Y_n^Q = f_n^T(U^Q)$ and $E(f_C(Y_n^Q))$, and then sends $E(f_C(Y_n^Q))$ to S_i. S_i computes the intersection between Y_n^Q and Y_i^T instead of Y_i^Q and Y_i^T. According to Proposition 5.1, f_n^T is a random polynomial and its coefficients are distributed uniformly over $\mathcal{F}$. The

elements of $Y_n^Q = f_n^T(U^Q)$ are also distributed uniformly over $\mathcal{F}$ according to Lemma 5.1. In order to achieve a successful impersonation, S_i has to at least guess k elements of Y_n^Q. This probability is $\frac{C(N_Q,k)}{|\mathcal{F}|^k}$ as shown above, and can be negligible due to $|\mathcal{F}|$ being very large.

Since S_i knows the intersection between Y_n^Q and Y_i^T unconditionally, he is likely to generate Y_i^T of a large size to detect the elements of Y_n^Q. In step 2 of PRIVATE-MATCHING, the size parameter g is used to limit the maximum number of elements in Y_i^T to evade such an attack.

5) **About size parameter g.** In the proposed authentication schemes, though the server and the user both need to learn the intersection G, we employ a conditioned mutual PM, instead of an ordinary mutual PM, because an imposter may make use of the property of the latter to explore the elements of Y^T stored in the server. In an ordinary mutual PM, both parties will know the intersection of the two sets after the protocol is carried out. Thus the imposter may adopt the following strategy to explore the elements of Y^T: he takes N_Q elements from $\mathcal{F}$ to construct a test set Y^Q and then uses it to carry out the protocol with the server. At the end of the protocol, the imposter will learn some elements of Y^T from the intersection of Y^T and Y^Q. After enough attempts, he may gather up k elements of Y^T and then successfully impersonate the legitimate user. For instance, if $|\mathcal{F}| = 2^{64}$ and $N_Q = 1000$, it will take an adversary $2^{64}/1000 = 1.8 \times 10^{16}$ attempts to search full $\mathcal{F}$. However, if the imposter steals and cracks the smart card, he can easily get Y^T because he only needs to detect U^T instead of Y^T – the domain of the elements of U^T being much smaller than $\mathcal{F}$. For example, if $N_Q = 100$ and $|\mathcal{I}| = 4000$, it will at most take him $4000/100 = 40$ attempts to learn U^T. However, the conditioned mutual PM in the proposed protocol overcomes the drawback of an ordinary mutual PM because, in this case, the user learns the intersection set only after he has been verified by the server and accordingly, the imposter must guess at least k elements of Y^T or U^T at one attempt. Hence, his chance is the same as the case that in 1). Obviously, the bigger the N_Q is, the fewer attempts the imposter needs. Therefore, in step 2 of PRIVATE-MATCHING, the size parameter, g, is used to set the upper limit of N_Q. Since the number of minutiae of a fingerprint is usually $20 \sim 100$ [44], g can be chosen a little larger than 100, e.g., 120.

6) **Session key.** When the authentication process is accomplished, the server and the user both know the intersection G, which is actually the shared secret between the two parties. Compared with an ordinary shared secret in cryptography, G will vary at each session automatically. This is due to the fact that the minutiae often vary when extracted from different fresh samples, and the order of elements in G may change at each authentication process (step 7 of CK-AUTHENTICATION). If a session key can be derived from G, there is no need to implement the session key exchange protocol. However, the randomness of $H(G)$ is limited, so it cannot be directly treated as the session key. Fortunately, v chosen by the server is a random number (step 2 of CK-AUTHENTICATION) and shared by two parties, so the two parties can compute $H(G||v)$ as the session key.

6. Experimental Results.

In the proposed protocol, there is a comparison between the transformed query set Y^Q and the reference set Y^T. As discussed in Section 3, for a large $|\mathcal{F}|$ (e.g., $|\mathcal{F}| = 2^{64}$), matching Y^Q against Y^T is essentially the equivalent of matching query U^Q against template U^T. Hence, we evaluate the accuracy performance of the proposed authentication scheme by directly matching U^Q against U^T without actually performing the protocol. The scheme is tested on FVC2002-DB2 fingerprint database, which consists of 800 fingerprint images from 100 fingers (eight impressions per finger). The size of an image is 296×560 pixels. The approach to biometric verification in our scheme is the same as that in [36], in which the query minutiae are compared directly

with the template and a threshold k is used to judge whether they are matched. We adopt the similar experimental scheme of [36]. Among eight impressions of each finger in FVC2002-DB2, we only use four high-quality impressions (impressions 1, 2, 7, and 8). In our experiments, we chose impression 1 as the template, and impressions 2, 7, and 8 as the queries. Thus, we had 100 templates and 300 queries. The main factors that affect the verification accuracy of the proposed scheme are alignment and quantization of the template and the query minutia sets. As stated in Section 3, alignment is a common problem in fingerprint-based bio-cryptosystems. Though a number of approaches have been presented, it is still an open problem. In this paper, we only focus on the effects of quantization. The minutiae of each impression were extracted and the matched minutia pairs were found by using the algorithm of [45], then quantized into U^Ts and U^Qs.

We use genuine accept rate (GAR) and false accept rate (FAR) to evaluate the verification accuracy of the scheme. The GAR is defined as the percentage of attempts made by genuine users such that $|U^Q \cap U^T| \geq k$. The number of genuine attempts is 300. The FAR is defined as the percentage of attempts made by an impostor such that $|U^Q \cap U^T| \geq k$, where U^Q is an impostor's quantized query. This corresponds to the case in which an impostor steals the smart card and uses his fingerprint to impersonate the legitimate user's fingerprint (see Section 5.2). The number of impostor's attempts is $99 \times 100 = 9900$. In any biometric system, a good performance means a high GAR and a low FAR. The compromise between GAR and FAR in our scheme can be controlled by the security parameter k.

For convenience, the quantization parameters q_x (indicating the quantized step along x of a minutia) and q_y (indicating the quantized step along y of a minutia) are both set to the same q in the experiments. Figure 3 illustrates the receiver operating characteristic (ROC) of the proposed scheme with respect to q at fixed k and $q_\theta = 30°$, in which the solid curve represents the case of $k = 10$ while the dotted curve represents the case of $k = 13$. Table 1 shows the GARs and FARs of the scheme with respect to k at fixed q and q_θ. In the table, two values of q, quantization 1 ($q = 26$ pixels) and quantization 2 ($q = 22$ pixels), are considered, while q_θ is equal to 30° in both cases.

From Table 1 and Figure 3, we can see that GAR and FAR both decrease with the increase of k, and they both increase with the increase of q. A bigger k leads to a higher security but a lower performance. Therefore, there should be a tradeoff between the two.

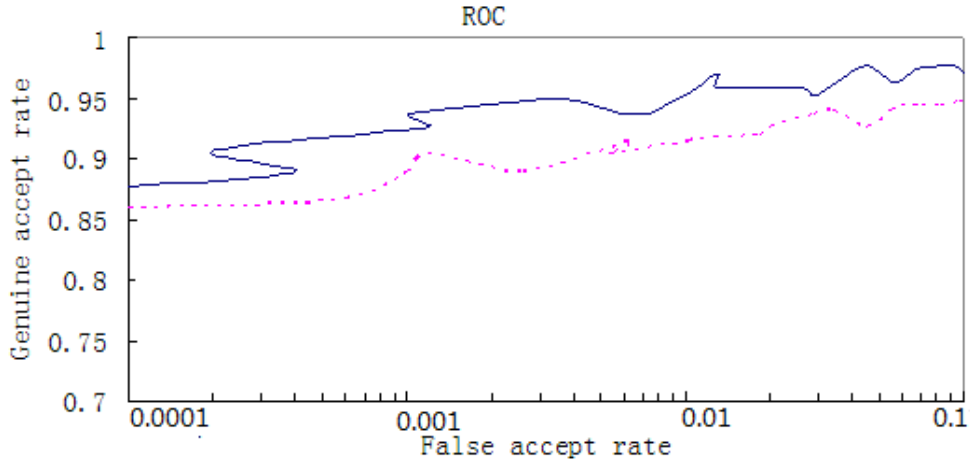

FIGURE 3. The ROC curve of the proposed scheme: The solid curve and dotted curve show the performances at $k = 10$ and of $k = 13$, respectively. $q_\theta = 30°$ in both cases.

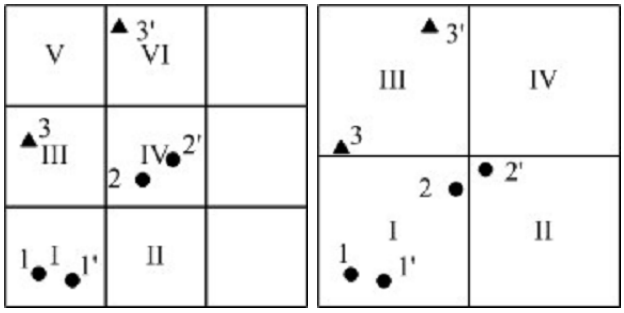

FIGURE 4. Example of fluctuations of ROC curve introduced by quantization. (a) Subspaces of quantization step q_1 and (b) subspaces of quantization step q_2.

TABLE 1. Performance of the proposed scheme on FVC2002-DB2 at fixed quantization parameters

Condition	k	8	9	10	11	12	13
Quantization 1	GAR (%)	96.7	95.7	93.7	91.7	89.7	86
$q = 26$ pixels	FAR (%)	0.54	0.16	0.12	0.04	0	0
Quantization 2	GAR (%)	96.0	93.7	90.7	89.3	86.3	81.7
$q = 22$ pixels	FAR (%)	0.20	0.08	0.02	0	0	0

In practice, a commonly accepted guideline is that two fingerprints are considered to be from the same finger as long as they share at least 12 minutia points. This guideline corresponds to $k = 12$ in the proposed scheme. From Table 1, we obtain GAR = 89.7% and FAR = 0. This level of accuracy is acceptable in practical use. Considering that the highest value of k used in [36] is 10 and its corresponding results are GAR = 86%, FAR = 0 at $k = 10$, we can say that our scheme shows better performance.

In Figure 3, the ROC curve shows some fluctuations, which are introduced by the quantization effect. Figure 4 gives an example to demonstrate how these fluctuations happen with respect to quantization. In both Figures 4(a) and 4(b), 1-1' and 2-2' are matched pairs while 3-3' is an unmatched pair. In Figure 4(a), 1 and 1' fall into I, 2 and 2' into IV, 3 into III, 3' into VI. Then 1-1', and 2-2' both are correctly matched, while 3-3' is unmatched. Though GAR increases with respect to quantization step q as a whole, at some specific q, GAR may decrease. For example, in Figure 4(b), 2' falls into II and 2 falls into I and thus a false un-matching occurs, while 3 and 3' both fall into III and accordingly a false matching occurs. As a result, the ROC curve fluctuates at q_2.

7. **Conclusion.** Biometric-based authentication gains advantages over password and token-based authentication. However, before it can be used in open networks, the privacy and security concerns that biometrics raise must be addressed. We provide a solution for privacy-preservation and secure authentication based on fingerprints over insecure channels. In the proposed scheme, the storing, transferring and matching of finger minutiae are all in encrypted or random domains, thus providing privacy protection, security and revocability. Both a server and a user are permitted to perform minutia matching to verify each other without revealing their respective private data. The proposed protocols

can withstand various attacks, such as a MITM attack, a replay attack, a collusion attack, and an impersonation attack; they also are inherently superior in performance and efficiency to other existing set-matching schemes, such as the fuzzy vault scheme.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (61062012, 90920001), the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] W. Stallings *Network Security Essentials: Applications and Standards (2/E)*, Prentice Hall, Upper Saddle River, NJ, 2003.
[2] C. Roberts, *Biometrics*, www.ccip.govt.nz/newsroom/information-notes/2005/biometrics.pdf, 2005.
[3] J. K. Lee, S. R. Ryu and K. Y. Yoo, Fingerprint-based remote user authentication scheme using smart cards, *Electronics Letters*, vol.38, no.12, pp.554-555, 2002.
[4] C. H. Lin and Y. Y. Lai, A flexible biometrics remote user authentication scheme, *Computer Standards & Interfaces*, vol.27, no.1, pp.19-23, 2004.
[5] M. K. Khan and J. S. Zhang, Improving the security of a flexible biometrics remote user authentication scheme, *Computer Standards & Interfaces*, vol.29, no.1, pp.82-85, 2007.
[6] J. Xu, W. T. Zhu and D. G. Feng, Improvement of a fingerprint-based remote user authentication scheme, *International Journal of Security and Its Apllications*, vol.2, no.3, pp.73-80, 2008.
[7] I.-S. Jeon, H.-S. Kim and M.-S. Kim, Enhanced biometrics-based remote user authentication scheme using smart cards, *Journal of Security Engineering*, vol.8, no.2, pp.237-256, 2011.
[8] U. Uludag, S. Pankanti, S. Prabhakar and A. K. Jain, Biometric cryptsystem: Issues and challenges, *Proc. of IEEE*, vol.92, no.6, pp.948-960, 2004.
[9] A. Jain, K. Nandakumar and A. Nagar, Biometric template security, *EURASIP Journal on Advances in Signal Processing*, pp.1-17, 2008.
[10] A. Juels and M. Wattenberg, A fuzzy commitment scheme, *ACM Conf. on Computer and Communications Security*, pp.28-36, 1999.
[11] A. Juels and M. Sudan, A fuzzy vault scheme, *Proc. of IEEE Int. Symp. Information Theory*, pp.408, 2002.
[12] Y. Dodis, L. Reyzin and A. Smith, Fuzzy extractors: How to generate strong keys from biometrics and other noisy data, *Eurocrypt*, Interlaken, Switzerland, pp.523-540, 2004.
[13] Y. Dodis, R. Ostrovsky, L. Reyzin and A. Smith, Fuzzy extractors: How to generate strong keys from biometrics and other noisy data, *SIAM Journal on Computing*, vol.38, no.1, pp.97-139, 2008.
[14] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky and A. Smith, Secure remote authentication using biometric data, *Proc of Eurocrypt*, Aarhus, Denmark, pp.147-163, 2005.
[15] S. Kanade, D. Petrovska-Delacretaz and B. Dorizzi, Generating and sharing biometrics based session keys for secure cryptographic applications, *The 4th IEEE International Conference on Biometrics: Theory Applications and Systems*, pp.1-7, 2010.
[16] F. Zhang and D. G. Feng, Fuzzy extractor based remote mutual biometric authentication, *Journal of Computer Research and Development*, vol.46, no.5, pp.850-856, 2009.
[17] K. Xi, T. Ahmad, F. Han and J. Hu, A fingerprint based bio-cryptographic security protocol designed for client-server authentication in mobile computing environment, *Security and Communication Networks*, vol 4, no.3, pp.487-499, 2011.
[18] Q. Tang, J. Bringer, H. Chabanne and D. Pointcheval, A formal study of the privacy concerns in biometric-based remote authentication schemes, *Proc. of the 4th Information Security Practice and Experience Conference*, Sydney, Australia, pp.56-70, 2008.
[19] N. D. Sarier, A new approach for biometric template security and remote authentication, *Advances in Biometrics – ICB, LNCS*, vol.5558, pp.916-925, 2009.
[20] M. Upmanyu, A. M. Namboodiri, K. Srinathan and C. V. Jawaha, Blind authentication: A secure crypto-biometric verification protocol, *IEEE Trans. on Information Forensics and Security*, vol.5, no.2, pp.255-268, 2010.
[21] A. Sadeghi, T. Schneider and I. Wehrenberg, Efficient privacy preserving face recognition, *Proc. of the 12th Annual International Conference on Information Security and Cryptology, LNCS*, vol.5984, pp.235-253, 2009.

[22] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk and T. Toft, Privacy-preserving face recognition, *Proc. of the 9th International Symposium on Privacy Enhancing Technologies*, pp.235-253, 2009.
[23] M. Barni, T. Bianchi, D. Catalano, M. D. Raimondo, R. D. Labati, P. Faillia, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti and A. Piva, Privacy-preserving fingercode authentication, *The 12th ACM Multimedia and Security Workshop*, pp.9-12, 2010.
[24] M. Barni, D. Catalano, M. D Raimondo, R. D Labati, P. Failla and T. Bianchi, A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates, *IEEE International Conference on Biometrics: Theory, Applications and Systems*, pp.27-29, 2010.
[25] M. Blanton and P. Gasti, *Secure and Efficient Protocols for Iris and Fingerprint Identification*, http://eprint.iacr.org/2010/627.pdf, 2010.
[26] Y. Huang, L. Malka, D. Evans and J. Katz, Efficient privacy-preserving biometric identification, *The 18th Network and Distributed System Security Conference*, pp.6-9, 2011.
[27] L. Hong, *Automatic Personal Identification Using Fingerprints*, Ph.D. Thesis, Michigan State University, 1998.
[28] S. Prabhakar, *Fingerprint Classification Matching Using a Filterbank*, Ph.D. Thesis, Michigan State University, 2001.
[29] R. Cappelli, D. Maio, D. Maltoni, J. L. Wayman and A. K. Jain, Performance evaluation of fingerprint verification systems, *IEEE Trans. on Pattern Analysis Machine Intelligence*, vol.28, no.1, pp.3-18, 2006.
[30] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, *Advances in Cryptology – EUROCRYPT*, pp.223-238, 1999.
[31] J. Benaloh, Dense probabilistic encryption, *Proc. of the Workshop on Selected Areas of Cryptography*, pp.120-128, 1994.
[32] M. Freedman, K. Nissim and B. Pinkas, Efficient private matching and set intersection, *Eurocrypt*, pp.1-19, 2004.
[33] R. Agrawal, A. Evfimievski and R. Srikant, Information sharing across private databases, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, San Diego, CA, 2003.
[34] L. Kissner and D. Song, Privacy-preserving set operations, *Advances in Cryptology, LNCS*, Berlin Heidelberg, vol.3621, pp.241-257, 2005.
[35] I. M. Verbauwhede and S. Yang, Secure fuzzy vault based fingerprint verification system, *Asilomar Conf. on Signals, Systems and Computers*, vol.1, pp.577-581, 2004.
[36] K. Nandakumar, A. K. Jain and S. Pankanti, Fingerprint-based fuzzy vault: Implementation and performance, *IEEE Trans. on Information Forensics and Security*, vol.2, no.4, pp.744-757, 2007.
[37] N. K. Ratha, S. Chikkerur, J. H. Connell and R. M. Bolle, Generating cancelable fingerprint templates, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.29, no.4, pp.561-572, 2007.
[38] J. Jeffers and A. Arakala, Fingerprint alignment for a minutiae-based fuzzy vault, *Biometrics Symposium*, pp.1-6, 2007.
[39] J. Li, X. Yang, J. Tian, P. Shi and P. Li, Topological structure-based alignment for fingerprint fuzzy vault, *Int. Conf. on Pattern Recognition*, Tampa, Florida, USA, pp.1-4, 2008.
[40] B. Toth, Biometric liveness detection, *Information Security Bulletin*, vol.10, pp.291-297, 2005.
[41] O. Goldreich, *Secure Multi-party Computation*, http://www.wisdom.weizmann.ac.il/∼oded/pp.html.
[42] ITU-T X.509 Recommendation, *Information Technology – Open Systems Interconnection – The Directory Public Key and Attribute Certificate Frameworks*, http://www.itu.int/rec/T-REC-X.509-200508-I, 2005.
[43] V. Shoup, *A Computational Introduction to Number Theory*, Version 1, http://www.shoup.net/ntb/.
[44] Y. F. Zhu, S. C. Dass and A. K. Jain, Statistical models for assessing the individuality of fingerprints, *IEEE Trans. on Information Forensics and Security*, vol.2, no.3, pp.391-401, 2007.
[45] X. H. Xie, F. Su and A. N. Cai, A robust fingerprint minutiae matching algorithm based on the support model, *The 1st Int. Conf. on Biometric Authentication*, Hong Kong, China, pp.316-323, 2004.