




Secure and Efficient Online Fingerprint Authentication Scheme Based On Cloud Computing

Yao Liu , Tanping Zhou , Zelun Yue, Wenchao Liu, Yiliang Han, Qi Li , and Xiaoyuan Yang

Abstract—Privacy protection of biometrics-based on cloud computing is attracting increasing attention. In 2018, Zhu *et al.* proposed an efficient and privacy-preserving online fingerprint authentication scheme for data outsourcing e-Finga. Under the premise of ensuring user's fingerprint data privacy and message security authentication, the e-Finga scheme can provide accurate and efficient fingerprint identity authentication services. However, our analysis shows that the temporary fingerprint in this scheme uses the deterministic encryption algorithm, which has the risk of leaking the user's fingerprint characteristics. Therefore, we propose a temporary fingerprint attack method for the e-Finga scheme. Experiments demonstrate that an adversary can analyze specific secret parameters and fingerprint features when eavesdropping on a user's temporary fingerprint ciphertext. To counter the temporary fingerprint attack, we propose a secure e-fingerprint scheme—Secure e-finger that uses the learning with errors samples, which has the homomorphic addition property, to encrypt user's temporary fingerprints. Experiments show that the secure e-finger scheme can resist the temporary fingerprint attack. Compared with the unprotected e-Finga scheme, the client running time is increased by about 6% percent, the communication cost on the user side only increased by 0.3125% percent. As a result, our solution can realize secure online fingerprint authentication without losing efficiency. Single user authentication is likely to cause the problem of excessive authority. Based on the Secure e-finger scheme, we propose a threshold scheme based on biological characteristics.

Index Terms—Homomorphic encryption, bilinear mapping, cloud computing, online authentication, fingerprint matching, threshold authentication

1 INTRODUCTION

WITH the rapid development of cloud computing technology and the popularization of mobile smart terminals, biometric authentication based on cloud computing has gradually penetrated every corner of people's daily lives [1]. A simple method of identifying people, biological characteristics, or biological behavior characteristics is receiving more attention for their convenience [2], [3].

However, while biometric authentication provides convenience to people's lives, it also brings about the privacy concerns. In 2019, the VpnMentor team attacked the security platform BioStar 2, leaving the fingerprint records and facial recognition information of more than one million users at risk of leakage [4]. **Once the biometric data is leaked, it cannot be revoked or replaced, and the same biometrics will be used in different systems [5].**

Therefore, it is important to establish appropriate security and privacy protection mechanisms to prevent fingerprint data leakage or abuse [6]. Many proposed biometric systems have been proposed using custom encryption and processing methods, leading to security risks. Considering the system's security and efficiency requirements, the privacy protection of online fingerprint authentication is still a challenging task.

In response to this problem, researchers have proposed a variety of solutions to solve it. In 1994, Bodo linked biometrics and cryptography for identification [7]. **To solve the contradiction between the cryptographic system's accuracy and the ambiguity of biometrics [8], [9], the Fuzzy Vault [10] scheme and the BioHashing [11] scheme propose corresponding solutions based on the error correction code and Hamming distance matching methods [12]. The BioHashing scheme has an obvious flaw. If an attacker pretends to be a legitimate user for identity authentication after obtaining the orthogonal matrix, there is a high probability that he can cheat the authentication system. The Fuzzy Vault scheme has two serious security flaws: (1) The data of the original**

- Yao Liu is with the Institute of School of Cryptographic Engineering, Engineering University of PAP, Xi'an, Shaanxi 710086, China. E-mail: liuyaoly61@foxmail.com.
- Tanping Zhou is with the TCA Laboratory, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China, and also with the Key Laboratory of Network and Information Security of PAP, Engineering University of PAP, Xi'an, Shaanxi 710086, China. E-mail: tanping2020@iscas.ac.cn.
- Zelun Yue is with the Department of Secrecy, Archive and Screen Logistics University of People's Armed Police Force, Tianjin 300300, China. E-mail: yuezulun@163.com.
- Wenchao Liu, Yiliang Han, and Xiaoyuan Yang are with the Key Laboratory of Network and Information Security of PAP, Engineering University of PAP, Xi'an, Shaanxi 710086, China. E-mail: liuwch3@mail2.sysu.edu.cn, hanyil@163.com, yangxiaoyly@126.com.
- Qi Li is with the School of Computer Science, University of Guelph, Guelph, ON N1G2W1, Canada. E-mail: qli15@uoguelph.ca.

Manuscript received 4 January 2021; revised 4 August 2021; accepted 5 August 2021. Date of publication 10 August 2021; date of current version 8 March 2023. (Corresponding author: Yao Liu.)

Recommended for acceptance by V. Piuri.

Digital Object Identifier no. 10.1109/TCC.2021.3103546

feature point template can be obtained by cross-comparing multiple fuzzy vaults; (2) Once the key is stolen, the attacker can replace part of the random hash value with another value, then can pass system verification by impersonating a legitimate user by verifying these values. In 2015, Blanton *et al.* proposed the FingerCode fingerprint authentication scheme [13] based on the homomorphic encryption scheme [14], which realized the privacy protection of fingerprint data in the two-party interaction scenario without affecting the authentication accuracy of the FingerCode fingerprint identification scheme. However, the FingerCode fingerprint authentication scheme has defects of high computational complexity and extensive communication cost. In 2015, Mohammad *et al.* [15] proposed an efficient privacy protection biometric identification solution CloudID using searchable encryption technology [16]. CloudID can perform biometric operations in an encrypted domain to ensure that cloud servers or potential attackers cannot directly access sensitive data. However, this scheme cannot provide complete privacy protection for the biometric authentication system, nor does it support the complex data ciphertext calculation in the biometric authentication system. To solve the problem of secure calculation of Hamming distance in biometric recognition, Masaya *et al.* [17] proposed secure conjunctive keyword search based on two homomorphic encryption schemes in 2017, which realized more complex data ciphertext calculation. However, the scheme has the defects of large communication volume and high computational complexity. To solve the system's requirements for security and efficiency, Zhu *et al.* proposed an efficient and privacy-preserving online fingerprint authentication scheme over Outsourced data, e-Finga [18], based on the semi-homomorphic encryption BGN scheme [19] in 2018. In the proposed e-Finga scheme, the user's fingerprint registered in the trust authority center can be outsourced to different servers with user's authorization, and it can provide secure, accurate, and efficient authentication service without the leakage of fingerprint information. The author verified the accuracy and efficiency of the scheme through experiments.

In this paper, we analyze the e-Finga scheme's security and designs a more secure and efficient secure e-finger scheme.

First, this paper analyzes the three security risks of the literature [18]: ① deterministic encryption algorithms encrypt temporary fingerprints, ② all the users use the same secret parameters, ③ the authorization data package does not authenticate the identity of the authorized object in the Encrypted Template Authorization phase. This paper proposes a temporary fingerprint attack to get user's fingerprints in response to the above risks. Our experiments show that we can get the fingerprint characteristics and some secret parameters in eavesdropping on the user's temporary fingerprint ciphertext.

Second, this paper proposes a secure online fingerprint authentication scheme Secure e-finger scheme. We use hard samples on the lattice cryptography to encrypt temporary fingerprints in the Authentication Query Generation phase. This approach has two benefits: on the one hand, the temporary fingerprint feature distribution information can be hidden; on the other hand, in the Fingerprint Matching phase,

the Euclidean distance between the template and the temporary fingerprint feature can be calculated without affecting the matching result. Secondly, different secret parameters are generated for different users to prevent collusion between corrupt users and servers who want to obtain secret parameters and legitimate user's template information. Finally, in the Encrypted Template Authorization phase, the authorization object information is added to the authorization data package. The method can prevent attackers from pretending to be authorized objects and pretending to be the user's fingerprint template data, then protecting the template data from abuse.

Finally, we have compared the proposed scheme and the e-Finga scheme from the aspects of the client running time, computational complexity, space cost, and communication cost, and the analysis showed that the security of the scheme could be improved without affecting the system operating efficiency.

This paper analyzes and discusses the literature [18]'s security issues and introduces the samples of LWE (Learning with errors) to encrypt the temporary fingerprint feature. This paper proposes a new security privacy protection online fingerprint authentication scheme based on cloud computing, called Secure e-finger, based on BGN homomorphic encryption scheme. In our scheme, the user's fingerprint can be outsourced to different servers with user's authorization, and it can provide a secure, accurate authentication service without the leakage of fingerprint information. The computation and the communication cost of our scheme are low.

2 PRELIMINARIES

This section introduces the system model of the Secure e-finger scheme, its design requirements, and the basic knowledge of cryptography related to the scheme, and introduction to the e-Finga scheme.

2.1 System Model and Design Requirements

This subsection introduces the system model of the Secure e-finger scheme. In the case of a third-party trusted authority, it can realize online fingerprint authentication services between users and online authentication service providers based on cloud computing, ensuring the privacy of user's fingerprint information and achieving secure identity authentication. Meanwhile, for easier expression, we give the description of variables used in the following subsections in Table 1.

2.1.1 System Model

The scheme in this article mainly includes three types of parties: trusted authority center (TA), Online Authentication Service Based on Cloud Computing (OAS), and users (U), as shown in Fig. 1.

Part 1: *Trusted Authority Center (TA)*: TA is a trusted third-party authority whose main functions are: system initialization, user registration, service provider registration, fingerprint template generation, storage, and authorization. After the TA system is initialized,

TABLE 1
Variables and Their Descriptions

Variables	Description
l, g, h, q_1, q_2	parameters of bilinear groups
G, G_T	the bilinear groups with order N
SB, PB	$SB = g^{q_1}, PB = e(g, g)^{q_1}$
sk_{TA}, pk_{TA}	the private key and public key of TA
$E()$	the secure asymmetric encryption algorithm
$H_1(), H_2()$	the secure cryptographic hash function
sk_s, pk_s	the private key and public key of OAS
IC_S	a secret identification code assigned by TA
SB_{U_i}, PB_{U_i}	The secure parameter of U_i
k_i, k'_i	random masking parameters
x_{U_i}, Y_{U_i}	the fingerprint characteristics of U_i
F_{U_i}	the template data of U_i
RQ_{U_i}	The authentication query data of U_i
RDS_i	the reference evaluation data set for U_i
$t^{(1)}, \dots, t^{(l)}$	l LWE samples generated by TA
$g^{q_1 \cdot k}$	The secret of the threshold scheme
$s(i)$	The shares of the threshold scheme

TA generates system parameters, send them the system parameters when users and online authentication service provider are registered in the center; Encrypt the user's fingerprint characteristics to generate a fingerprint template; and Provide encrypted User's fingerprint template information to the OAS, when it requests a template.

Part 2: *Online Authentication Service Provider Based on Cloud Computing (OAS)*: Main functions: accept user's service requests, apply fingerprint templates, secret fingerprint matching, and complete authentication requests. OAS should be registered to TA; when a user makes a service request to the OAS, OAS will apply to TA for the user's fingerprint template information. The temporary fingerprint ciphertext submitted by the user is secretly matched with the template information and returns the matching result to the user.

Part 3: *User (U)*: Main functions: collect and upload fingerprint characteristics, apply for services from an online authentication service provider, collect and upload temporary fingerprint ciphertext. The user registers at the TA and collects and uploads fingerprint characteristics; the user makes a service request to the OAS; the user collects the fingerprint characteristics, generates a temporary fingerprint ciphertext, and uploads it to the OAS.

The four processes of the entire system model are as follows:

Phase 1: In the System Initialization phase, TA generates the system parameters required by the system. The user and service provider sends a registration request to the TA. The TA returns the system initialization parameters; the user collects and uploads fingerprint characteristics to the TA, and the TA generates a fingerprint template after encryption.

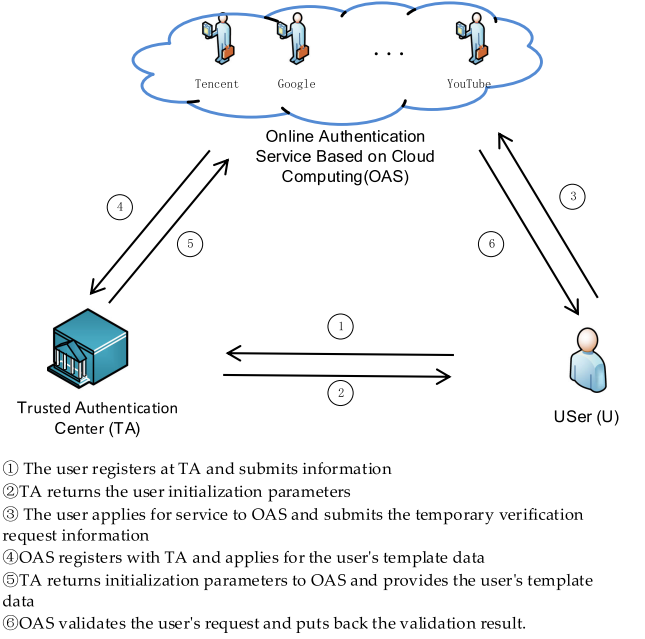


Fig. 1. System model.

Phase 2: In the Encrypted Template Authorization phase, the user makes a service request to the OAS. The OAS applies to the TA for the user's template data. TA sends the user's template information to the service provider for fingerprint matching.

Phase 3: In the Authentication Query Generation phase, the user collects fingerprint characteristics, encrypts the temporary fingerprint ciphertext, and submits it to the service provider for authentication.

Phase 4: In the Fingerprint Matching phase, the service provider performs secret state matching between the temporary fingerprint ciphertext submitted by the user and the template information and returns the matching result to the user.

2.1.2 Design Requirements

Design requirements follow the literature [18]. The scheme in this article mainly includes three types of parties: TA, OAS and U. TA is a trusted third party, such as a government or a reputable agency. OAS is semi-trusted. While completing the function of identity authentication, it also tries to analyze template data and query data to obtain the original fingerprint information. In addition, OAS may pretend to be other OAS to provide identity authentication. We assume that there is an active opponent C in the communication process, which can obtain the communication data in the channel, guess the plaintext value of the data and impersonate a legitimate user or OAS. The focus of the identity authentication scheme lies in the security of fingerprint data in the storage, transmission and calculation process. Therefore, the online fingerprint identity authentication scheme needs to meet the following security requirements.

Req 1: *Privacy*: The scheme should protect the user's fingerprint information. Even if OAS or attacker C obtains

the user's identity authentication request information, the original fingerprint information cannot be obtained from it. When OAS matches the request data with the template data, it cannot obtain any useful information other than the matching result.

Req 2: *Confidentiality*: The Secure e-finger scheme needs to ensure the confidentiality of the fingerprint template. Even if OAS stores the template data of the user authorized by the TA, it cannot obtain the original fingerprint information from template data. Attacker C monitors the communication channels of the three parties and obtains the template data in the channel, but still cannot forge the template data. An illegal OAS obtains the corresponding template data from another OAS, and it cannot perform fingerprint authentication.

Req 3: *Authentication*: The most important point is to authenticate the identity authentication request which the user send to the legitimate OAS. OAS can calculate the Euclidean distance in the ciphertext state. If an illegal user forges a query or response, this malicious operation should be detected. Only legitimate queries and responses can be accepted.

2.2 Background

This section explains our scheme's basic knowledge, bilinear pairing technology, and BGN cryptosystem and then describes the identity matching algorithm based on the fingerprint code, which is the basic content of our scheme.

(1) Bilinear mapping [20]

Give two cyclic groups of the same finite order n : G and G_T . The g is a generator of G . There is a bilinear map e between G and G_T . The bilinear map has the following properties.

Bilinear: There must be $e(u^a, v^b) = e(u, v)^{ab}$, where $u, v \in G$ and $\forall a, b \in \mathbb{Z}_q^*$;

Non-degeneracy: $e(g, g) \neq 1$;

Computability: For any $u, v \in G$, the value of $e(u, v)$ can be calculated efficiently.

(2) BGN cryptosystem [19]

The BGN cryptosystem is a homomorphic encryption scheme proposed by Boneh, Goh, and Nissim in 2005. The algorithm consists of four parts: key generation, encryption, decryption, and homomorphic evaluation.

Part 1: Key Generation KeyGen(l): Given a security parameter $\lambda \in \mathbb{Z}^+$, generates a tuple (q_1, q_2, G, G_T, e) , where q_1 and q_2 are two distinct large primes, G is a cyclic group of order $q_1 q_2$, and e is a pairing map $e: G \times G \rightarrow G_T$. Let $N = q_1 q_2$. Pick up two random generators g, u from G and set $h = u^{q_2}$. Then h is a random generator of the subgroup of G with order q_1 . The public key is $pk = \{N, G, G_T, e, g, h\}$. The secret key is $sk = q_1$.

Part 2: Encryption Enc_{pk}(m): Assumed plaintext space $m \in \{0, 1, \dots, T\}$, $T < q_2$. Choose a random number $r \in \{0, 1, \dots, T\}$, encrypt the plaintext message m with the public key pk , and get the ciphertext $C = g^m \cdot h^r \in \mathbb{G}$.

Part 3: Decryption Dec_{sk}(C): Input a ciphertext C , a security key $sk = q_1$, compute $C^{q_1} = (g^m \cdot h^r)^{q_1} = g^{mq_1} = (g^{q_1})^m$, and set $\hat{g} = g^{q_1}$. To achieve the corresponding message m , it suffices to compute the discrete logarithm of C^{q_1} with \hat{g} , which takes the expected time $O(\sqrt{T})$ using Pollard's lambda method, where $0 \leq m \leq T$.

Part 4: Homomorphic evaluation: Homomorphic addition HomAdd(C_1, C_2): Given two ciphertexts $C_1 = g^{m_1} h^{r_1} \in G$, $C_2 = g^{m_2} h^{r_2} \in G$ of messages $m_1, m_2 \in \{0, 1, \dots, T\}$, pick a random $r \in \mathbb{Z}_N$, computing the product $C_+ = C_1 C_2 h^r$.

Homomorphic multiplication HomMult(C_1, C_2): Given two ciphertexts $C_1 = g^{m_1} h^{r_1} \in G$, $C_2 = g^{m_2} h^{r_2} \in G$ of messages $m_1, m_2 \in \{0, 1, \dots, T\}$, pick a random $r \in \mathbb{Z}_N$ $h_1 = e(g, h)$, computing the product $C = e(C_1, C_2) h_1^r \in G_T$.

(3) LWE problem [21]

Learning with errors (LWE) problem is a common tool in lattice cryptography, and its security can be reduced to hard problems on lattices. The cryptography based on the LWE problem has the advantages of high computational efficiency and high security, and has been widely studied.

Definition (LWE). For the security parameter λ , let $n = n(\lambda)$ be an integer dimension, let $q = q(\lambda) \geq 2$ be an integer, let $\chi = \chi(\lambda)$ be a distribution over \mathbb{Z} . The $LWE_{n,q,\chi}$ problem is to distinguish the following two distributions.

Choose $e_i \leftarrow \chi$, $s \leftarrow \mathbb{Z}_q^n$, $a \leftarrow \mathbb{Z}_q^n$, calculation $b = \langle a, s \rangle + e$, output $(a, b) \in \mathbb{Z}_q^{n+1}$. Let the distribution be $A_{q,s,\chi}$. $LWE_{n,q,s}$ problem is distinction $A_{q,s,\chi}$ distribution and evenly distribution on \mathbb{Z}_q^{n+1} .

(4) Fingerprint data matching

After the image and vector extraction of Gabor filters, the FingerCode of a fingerprint can be generated as an n -dimensional feature vector, and each element is an 8-bit integer. We calculate the Euclidean distance between two fingerprint codes to determine whether the two fingerprints are from the same person. For example, giving two fingerprint codes $\mathbf{x} = (x_1, x_2, \dots, x_n)$, $\mathbf{y} = (y_1, \dots, y_n)$, their Euclidean distance is

$$d_{xy} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$$

If the result of the Euclidean distance is less than the threshold value Δ'_d , the corresponding fingerprint can be considered to be from the same person. When the error rate is in the range of 3-5% percent, it has a much lower computational complexity. Meet the needs of online fingerprint authentication.

2.3 Introduction to the E-Fing Scheme

This section mainly introduces the operation process of the e-Fing scheme[18]. The scheme mainly includes four phases: System Initialization, Encrypted Template Authorization, Authentication Query Generation, Fingerprint Matching.

Phase 1: System Initialization

(1) TA initialization: TA runs function keygen (l), generates parameters $\langle G, G_T, e, q_1, q_2, g, h, N \rangle$,

generates public secret parameters for all users $SB = g^{q_1}$, $PB = e(g, g)^{q_1}$, sets threshold Δ_d . TA uses the above parameters to construct the bloom filter BF_{RDS} , generates masking parameters k_i for registered users i . After OAS registered, TA generates a secret ID number cs_j for OAS, definition a vector $IC_S = \{cs_1, \dots, cs_j, \dots\}$. For the convenience of the presentation, we will use c_s replace cs_j in the following description.

- (2) Generate fingerprint template: User gets parameters $\langle SB, PB, k_i, IC_S \rangle$ from TA, collects and uploads user's fingerprint feature vector $x_{U_i} = (x_1, x_2, \dots, x_n)$, where n is the vector dimension. TA uses the parameter $\langle PB, k_i, IC_S \rangle$ to encrypt the fingerprint feature vector to generate template data $F_{U_i} = (f_{x_1}, f_{x_2}, \dots, f_{x_n}, f'_x)$, where $f_{x_j} = g^{x_j} \cdot h^{r_j}$, where masking data $x'_j = x_j + H_2(k_i + c_s)$, $j \in \{1, \dots, n\}$, where random number $r_1, r_2, \dots, r_n \in Z_N^*$, where $f'_x = PB^{(x'_1{}^2 + x'_2{}^2 + \dots + x'_n{}^2)}$, where $H_2()$ is the hash function.

Phase 2: Encryption template authorization

- (1) User registers at the authentication provider: the user signs the user ID ID_{U_i} and timestamp TS_1 with his private key. After signing, send data $\langle ID_{U_i} || TS_1 || Sig_{U_i} \rangle$ to the fingerprint authentication server.
- (2) The OAS requests the TA's template: the OAS verifies the user's identity, uses Sig_{U_i} to generate template request information, and requests the template F_{U_i} from TA.
- (3) TA provides fingerprint templates to OAS: the TA verifies the requested information's authenticity and the OAS's identity and then returns the fingerprint template data and user ID.
- (4) OAS storage template: OAS verifies the authenticity of the TA's identity and the message's integrity and saves the data $\langle ID_{U_i} || F_{U_i} \rangle$.

Phase 3: Authentication Query Generation

- (1) User generates temporary fingerprint ciphertext: User i uses terminal equipment to collect his fingerprint image, obtains fingerprint feature vector $Y_{U_i} = (y_1, y_2, \dots, y_n)$, user's parameters $\langle SB, PB, k_i, IC_S, \Delta_d \rangle$ to encrypt fingerprint feature vector, and generates temporary fingerprint ciphertext $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_n}, rq'_y)$, where $rq_{y_j} = SB^{y_j}$, masking data $y'_j = y_j + H_2(k_i + c_s)$, $j \in \{1, \dots, n\}$, $rq'_y = PB^{(y'_1{}^2 + y'_2{}^2 + \dots + y'_n{}^2 - \Delta_d^2)}$.
- (2) The user uploads the temporary fingerprint ciphertext and user ID $\langle RQ_{U_i} || ID_{U_i} \rangle$ to the corresponding OAS.

Phase 4: Fingerprint Matching

- (1) Encrypted fingerprint matching: The OAS calculates the Euclidean distance $M_d = \frac{e(f_{x_1}, rq_{y_1}) \cdot e(f_{x_2}, rq_{y_2}) \cdots e(f_{x_n}, rq_{y_n})}{f'_x \cdot rq'_y}$ between the temporary fingerprint ciphertext RQ_{U_i} submitted by the user and template information F_{U_i} . The Euclidean distance M_d is calculated in the

encrypted state; M_d is input to the Bloom filter BF_{RDS} , and the matching result is obtained by judging whether the threshold is exceeded. If the result RS is true, the identity authentication is successful; otherwise, the identity authentication fails.

- (2) OAS returns the matching result RS to the user.

3 INSUFFICIENCY AND ATTACK EXPERIMENT

This section mainly analyzes and discusses the hidden safety hazards of the e-Finga scheme[18] in the operation process, gives the scheme's optimization direction designs an attack method, and conducts experimental verification.

3.1 Defects and Optimization of the E-Finga Scheme

The e-Finga scheme has the following three defects, and we propose optimization directions for each of them.

Defect 1: In the e-Finga scheme, the template data of the authority and the user's temporary fingerprint data are encrypted using the same secret parameters SB , PB , and all users encrypt their fingerprint information in the same group SB , PB . In actual application scenarios, the fingerprint authentication servers can obtain SB , PB by temporarily encrypting the fingerprint information multiple times. The leakage of public secret parameters system parameters will cause the system to be attacked.

Optimization 1: Optimization direction: In the Secure e-finger scheme of this paper, the TA will select a random parameter k'_i for each user and then generate a secret parameter SB , PB unique to each user, where $SB = g^{q_1}$, $PB = e(g, g)^{q_1}$. Different users use different secret parameter settings. Even if a corrupt user conducts a collusion attack with OAS, the secret parameter SB , PB of other users cannot be obtained, which can improve the scheme's security.

Defect 2: In the e-Finga scheme, when the user registers with the OAS, the data packet sent to the service provider is $\langle ID_{U_i} || TS_1 || Sig_{U_i} \rangle$, where $Sig_{U_i} = H_1(ID_{U_i} || TS_1)^{SK_{U_i}}$. The data package only contains the user's personal information ID_{U_i} , and does not contain the information of the service provider requested by the user. A malicious OAS attacker can intercept and capture the data packet $\langle ID_{U_i} || TS_1 || Sig_{U_i} \rangle$ in the user and the other normal OAS communication channel. Then the malicious OAS can use the data packet to request and successfully obtain the user template from the TA, which will cause the leakage of the user's fingerprint template information.

Optimization 2: Optimization direction: In our Secure e-finger scheme, the ID information of the OAS is added to the data packet when the user registers with the OAS. Our data packet consists $\langle ID_S || ID_{U_i} || TS_1 || Sig_{U_i} \rangle$, where $Sig_{U_i} = H_1(ID_S || ID_{U_i} || PK_S || TS_1)^{SK_{U_i}}$. The malicious OAS attacker cannot use the data packet of the user and other OAS to request the user's template information from the TA because the data packet contains user information ID_{U_i} and normal OAS information

ID_S , and the information cannot be changed due to the signature.

Defect 3: The e-Finga scheme cannot resist temporary fingerprint attacks. In the Authentication Query Generation phase, the same value $H_2(k_i + c_s)$ is added to the fingerprint features y_j of different dimensions of the user to obtain the masking data $y'_j = y_j + H_2(k_i + c_s)$. Then the masking data is encrypted to obtain $rq_{y_j} = SB^{2y'_j}$. In this way, the temporary fingerprint ciphertexts $rq_{y_j} = SB^{2y'_j}$ of the same user have little difference, which will expose the distribution information of fingerprint characteristics. That is, the two adjacent temporary fingerprint ciphertexts are relatively similar and have the same distribution. For example, let rq_{y_j} be the first temporary fingerprint ciphertext, and $rq_{y_{j+1}}$ be the second temporary fingerprint ciphertext, then $\frac{e(g, rq_{y_j})}{e(g, rq_{y_{j+1}})} = PB^{2(y_j - y_{j+1})}$, because the difference between the two fingerprint features $y_j - y_{j+1}$ is small, so PB^2 can be obtained from $\frac{e(g, rq_{y_j})}{e(g, rq_{y_{j+1}})}$. Further, the user's fingerprint characteristics y_j will be revealed. The privacy of user fingerprint features cannot be guaranteed.

Optimization 3: Optimization direction: In the Secure e-finger scheme of this article, the U uses the difficult instance on the grid to encrypt the temporary fingerprint during the Authentication Query Generation phase, thereby hides the distribution information of the feature. The temporary fingerprint masking data of each encryption has a massive difference, so it can be sufficient to prevent temporary fingerprint attacks. In the System Initialization phase, the system generates l LWE samples $t^{(1)}, \dots, t^{(l)}$ for users, where $t^{(j)} \in Z_N^{n+1}$, which satisfies the condition $\sum_{i=1}^{n+1} t_i^{(j)} x'_i = e_j \bmod N$, n is the dimension of the feature vector, and e_j obeys the discrete Gaussian distribution. In the Authentication Query Generation phase, the U only needs to randomly select a small coefficient linear combination of $t^{(1)}, \dots, t^{(l)}$ to obtain t , and calculate $rq_{y_i} = SB^{2y'_i + t_i}$ during encryption. The LWE instance t can protect the user's temporary fingerprint y_j well, and the x'_i in the template can be used during verification to eliminate the t in the temporary fingerprint to ensure the accuracy of the scheme. See section 3.2 for specific methods and attack experiments.

3.2 Design of Attack Method

The e-Finga scheme cannot resist temporary fingerprint attacks. When the user makes multiple service requests to the service provider, they need to upload multiple temporary fingerprint ciphertext information, and the service provider can obtain multiple temporary fingerprints of the user. After the ciphertext information, the user's fingerprint feature plaintext information can be analyzed. The specific attack method is designed as follows:

(1) Analyze temporary fingerprint ciphertext data to obtain secret parameters SB^2 , PB^2 .

When the e-Finga scheme generates the masking data, each fingerprint feature component y_j uses the same data $H_2(k_i + c_s)$. This processing method changes the fingerprint

feature component but does not disrupt the distribution information. We can attack it as following:

Input: User's two temporary fingerprint ciphertexts $RQ_{U_i}, \overline{RQ}_{U_i}$ which are encrypted by temporary fingerprint features Y_{U_i}, \bar{Y}_{U_i} .

$$RQ_{U_i} = \begin{cases} rq_{y_1} = SB^{2y'_1} \\ rq_{y_2} = SB^{2y'_2} \\ \vdots \\ rq_{y_n} = SB^{2y'_n} \\ rq'_y = PB^{y_1'^2 + y_2'^2 + \dots + y_n'^2 - \Delta_d^2} \end{cases}$$

$$\overline{RQ}_{U_i} = \begin{cases} \overline{rq}_{y_1} = SB^{2\bar{y}'_1} \\ \overline{rq}_{y_2} = SB^{2\bar{y}'_2} \\ \vdots \\ \overline{rq}_{y_n} = SB^{2\bar{y}'_n} \\ \overline{rq}'_y = PB^{\bar{y}_1'^2 + \bar{y}_2'^2 + \dots + \bar{y}_n'^2 - \Delta_d^2} \end{cases}$$

The attack process is as follows:

1) Obtained $d_{RQ_{U_i}}$ by calculating the ratio of the ciphertext feature components separately, the calculation formula is as follows:

$$d_{RQ_{U_i}} = \frac{RQ_{U_i}}{RQ_{U_i}} = \begin{cases} d_1 = SB^{2y'_1 - 2\bar{y}'_1} \\ d_2 = SB^{2y'_2 - 2\bar{y}'_2} \\ \vdots \\ d_n = SB^{2y'_n - 2\bar{y}'_n} \\ rq'_y = PB^{y_1'^2 + y_2'^2 + \dots + y_n'^2 - (\bar{y}_1'^2 + \bar{y}_2'^2 + \dots + \bar{y}_n'^2 - \Delta_d^2)} \end{cases}$$

Because $y'_j = y_j + H_2(k_i + c_s)\bar{y}'_j = \bar{y}_j + H_2(k_i + c_s)$, $j \in \{1, \dots, n\}$, so $2y'_j - 2\bar{y}'_j = 2y_j - 2\bar{y}_j$, usually the difference between the two fingerprint feature vectors is small.

2) For each component d_j of $d_{RQ_{U_i}}$, perform bilinear mapping to calculate the power of PB . $e(g, d_j) = e(g, SB^{2\Delta y_j}) = PB^{2\Delta y_j}$, $j \in \{1, \dots, n\}$, where $\Delta y_j = y_j - \bar{y}_j$.

3) Use Bloom filter to determine whether $e(g, d_j)$ is in BF_{RDS} , where $j \in \{1, \dots, n\}$. If $e(g, d_j)$ is in BF_{RDS} , then calculate the $1 - \lfloor \frac{\Delta_d^2}{2} \rfloor$ powers of d_j in turn; if's $1 - \lfloor \frac{\Delta_d^2}{2} \rfloor$ powers are all in BF_{RDS} , then d_j can be regarded as SB^2 and $e(g, d_j)$ is PB^2 , otherwise replace with a new temporary fingerprint and recalculate.

When $\Delta y_j = 1$, the attack process can get the correct SB^2 and PB^2 . Usually, the difference Δy_j between the two fingerprint feature vectors during temporary sampling is small. When the value range j is large enough, and there are enough temporary fingerprints, it must be There will be a case of $\Delta y_j = 1$ (experiments show that usually only two temporary fingerprints are needed to find the SB^2 and PB^2).

(2) Analyze temporary fingerprint ciphertext data to obtain redundant values $SB^{2H_2(k_i + c_s)}$

On the basis that SB^2 has been obtained, each component $rq_{y_j} = SB^{2y'_j} = SB^{2(y_j + H_2(k_i + c_s))}$ in the temporary fingerprint

ciphertext $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y'_j})$ ($j \in \{1, \dots, n\}$, y_j is 8 bits data) is divided by the 0-255 power of SB^2 to obtain the matrix $rqyd$, as shown in the following formula.

$$rqyd = \begin{bmatrix} \frac{rq_{y_1}}{SB^{2*0}} & \frac{rq_{y_1}}{SB^{2*1}} & \dots & \frac{rq_{y_1}}{SB^{2*255}} \\ \frac{rq_{y_2}}{SB^{2*0}} & \frac{rq_{y_2}}{SB^{2*1}} & \dots & \frac{rq_{y_2}}{SB^{2*255}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{rq_{y_n}}{SB^{2*0}} & \frac{rq_{y_n}}{SB^{2*1}} & \dots & \frac{rq_{y_n}}{SB^{2*255}} \end{bmatrix}$$

In each row of the matrix $rqyd$, the value $SB^{2H_2(k_i+c_s)}$ must be included. Find the intersection of n rows of data. If the intersection has only one value, the attack is successful, and this unique value is $SB^{2H_2(k_i+c_s)}$. If the data in the intersection is not unique, you can find the desired value by exhaustively constructing the temporary fingerprint ciphertext. The specific exhaustive method is as follows:

- Step 1:** Take any value in the set, assume it is $SB^{2H_2(k_i+c_s)}$, use this parameter and the temporary fingerprint feature Y_{U_i}' obtained by the method in Section 3.2 (3);
- Step 2:** Encrypt the temporary fingerprint feature Y_{U_i}' to obtain the temporary ciphertext RQ_{U_i}' ;
- Step 3:** Use the user's templates F_{U_i} and RQ_{U_i}' for ciphertext matching. If the ciphertext is matched successfully, the value is $SB^{2H_2(k_i+c_s)}$, if the match is unsuccessful, use the other elements in the intersection to perform exhaustive calculation, and repeat steps one to three.

The efficiency of the attack is related to the scale of the matrix $rqyd$. To reduce the number of rows in the matrix and improve the efficiency of the attack, this paper keeps only one of the same components in the temporary fingerprint ciphertext component, that is, removes duplicate rows in the matrix. The number of elements in the above intersection is related to the maximum difference $\max\{y_i - y_j\}_{i,j \in \{1, \dots, n\}}$ of the plaintext of the temporary fingerprint ciphertext component rq_{y_j} . To increase the diversity of the temporary fingerprint ciphertext component and reduce the number of elements in the intersection, this paper combines multiple temporary fingerprint ciphertexts of the same user. The text is cascaded to construct a larger-dimensional ciphertext. In the specific experiment, we use ten 100-dimensional temporary fingerprint ciphertexts of the same user to construct a 1000-dimensional temporary fingerprint ciphertext set for testing.

(3) Analyze temporary fingerprint ciphertext data to obtain fingerprint feature $Y_{U_i} = (y_1, y_2, \dots, y_n)$

Based on obtaining SB^2 and $SB^{2H_2(k_i+c_s)}$, calculate $RQY = (rq_0, rq_1, \dots, rq_{255})$, where $rq_j = SB^{2H_2(k_i+c_s)} \cdot (SB^2)^j$, $j \in \{0, \dots, 255\}$, as shown in Formula (1). Because y_j is 8 bits data, RQY is the ciphertext space of the temporary fingerprint ciphertext. Therefore, the component $rq_{y_j} = SB^{2y_j} = SB^{2H_2(k_i+c_s)} \cdot (SB^2)^{y_j}$ of the temporary fingerprint ciphertext $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_n}, rq_{y'_j})$ belongs to the temporary fingerprint ciphertext space RQY .

Compare rq_{y_j} with the set RQY , if $rq_{y_j} = rq_k$, then rq_{y_j} corresponds to the fingerprint feature plaintext as k . Further, the plain text $Y_{U_i} = (y_1, y_2, \dots, y_n)$ of all user fingerprint

characteristics can be found.

$$RQY = \begin{cases} rq_0 = SB^{2H_2(k_i+c_s)} \cdot (SB^2)^0 \\ rq_1 = SB^{2H_2(k_i+c_s)} \cdot (SB^2)^1 \\ rq_2 = SB^{2H_2(k_i+c_s)} \cdot (SB^2)^2 \\ \vdots \\ rq_{255} = SB^{2H_2(k_i+c_s)} \cdot (SB^2)^{255} \end{cases} \quad (1)$$

3.3 Attack Experiment

Experimental environment: The main system uses Java as development language, core i5-6300-2.3GHz quad-core processor, 16GB ram and windows 10 operating system. Fingerprint collector: ZKTeco Live20R Fingerprint Apparatus.

Preparation for the experiment: As an attacker, it can obtain verification information submitted by users in the communication channel. Use the fingerprint collector to collect the fingerprint of the same person 10 times. Get the fingerprint features Y_1, Y_2, \dots, Y_{10} , $Y_i = (y_{i,0}, y_{i,1}, \dots, y_{i,n-1})$, $i \in \{2, \dots, 10\}$, $j \in \{1, \dots, n-1\}$. Use the e-Finga scheme to obtain verification information $RQ_{U_{i,1}}, RQ_{U_{i,2}}, \dots, RQ_{U_{i,10}}$.

In the e-Finga scheme, the threshold Δ_d^2 of matching is given and construct the bloom filter BF_{RDS} , in which $RDS = \{RD_1, RD_2, \dots, RD_{\Delta_d^2}\}$, $RD_i = PB^i$, $i \in \{0, 1, \dots, \Delta_d^2\}$.

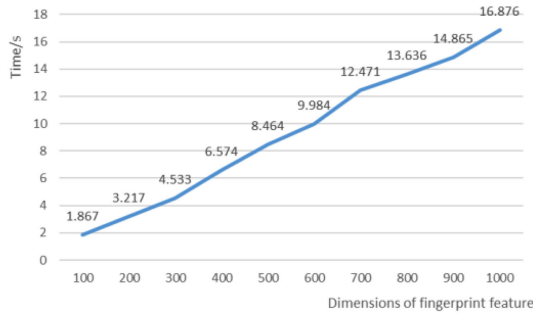
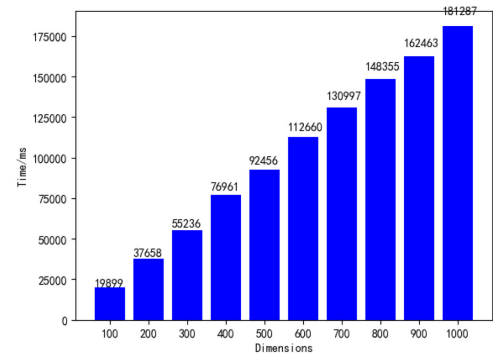
Experiment One: Try to obtain the SB^2, PB^2 data of the e-Finga scheme

There are three parties involved in the e-Finga scheme: users, online authentication service provider (OAS), and trusted authority center (TA). The OAS agency can apply for the user's encrypted fingerprint template information, and can also receive the user's temporary encrypted fingerprint information for verification. The same user may log in to the authentication server at different times, and therefore may send multiple service requests to the server. Collect the fingerprint feature ciphertext in multiple temporary requests, and use the method in Section 3.2 (1) to analyze the temporary fingerprint ciphertext data to obtain the secret system parameters SB^2, PB^2 .

The experiment inputs two temporary fingerprint ciphertexts of the same user, and the fingerprint feature dimension n is from 100 to 1000, and the attack effect and time are tested in 10 groups. Each group of 30 experiments can successfully analyze the secret parameter SB^2, PB^2 . The average time of each experiment is shown in Fig. 2. For 2 temporary fingerprint ciphertexts of 1000 dimensions, the time to recover the secret parameters does not exceed 15 seconds.

Experiment Two: Try to obtain the $SB^{2H_2(k_i+c_s)}$ data of the e-Finga scheme

Based on experiment 1, collect the fingerprint feature ciphertexts in multiple temporary requests, and divide the fingerprint feature dimension n from 100 to 1000 in 10 groups to test the attack effect and time. In each group of experiments, it is assumed that the encrypted ciphertext information $RQ_{U_i} = (rq_{y_{i,1}}, \dots, rq_{y_{i,j}}, \dots, rq_{y_{i,n}}, rq_{y'_j})$, which is encrypted from the characteristic of the 10th temporary fingerprint Y_1, \dots, Y_{10} is collected from the same user, in which $rq_{y_{i,j}}$ means the ciphertext corresponding to the i -th j -dimensional temporary fingerprint. Use the method in Section 3.2 (2) to analyze the temporary fingerprint

Fig. 2. The time to obtain SB^2 from 100 to 1000 dimensional.Fig. 3. Time of calculating plain text data y .

ciphertext data to obtain the redundancy value $SB^{2H_2(k_i+c_s)}$. For the same user ten times of 1000-dimensional temporary fingerprint feature ciphertext attacks, the average time to successfully obtain $SB^{2H_2(k_i+c_s)}$ is 49.9 seconds, and the successful time of the other groups of attacks is shown in the Table 2.

Experiment Three: Try to obtain the plaintext data y of the e-Fing scheme

Based on finding $SB^{2H_2(k_i+c_s)}$ and SB^2 in experiments one and two, the experiment inputs the user's temporary fingerprint ciphertext $RQ_{U_i} = (rq_{y1}, rq_{y2} \dots rq_{yn}, rq'_{y1})$. The fingerprints feature dimension n from 100 to 1000 is divided into ten groups to test the attack effect and time. Each group conducts 30 experiments, all it can successfully analyze the fingerprint characteristics $Y_{U_i} = (y_1, y_2 \dots y_n)$, and the average time of each experiment is shown in Fig. 3.

For a 1000-dimensional temporary fingerprint ciphertext, the average time to restore the plaintext of fingerprint features does not exceed 190 seconds.

4 SECURE E-FINGER SCHEME

To avoid those security risks, this paper proposes a novel fingerprint matching scheme Secure e-finger scheme. In the Secure e-finger scheme, TA assigns different encryption parameters to each user, uses the BGN homomorphic encryption algorithm to encrypt the user's fingerprint to generate template data; when the user is authenticated, it combines the LWE instance to privacy the user's temporary fingerprint protection. The Secure e-finger scheme has three parties involved: the users (U), online authentication service providers (OAS), and the trusted authority center (TA). The main process consists of four parts: System Initialization, Encrypted Template Authorization, Authentication Query Generation, Fingerprint Matching.

4.1 System Initialization

In our scheme, the trusted authority center (TA) is a safe and reliable third party. In real life, this trusted third party can be a government agency.

Part A: TA Initialization and Institution Registration

After system initialization, TA generates some system parameters. Users and the online authentication service provider will register in the system. After registration, users and OAS will have their own public and private keys to encrypt and authenticate three-party information transmission.

Phase 1: The TA initialization: TA selects a security parameter $l \in Z^+ (l > 512)$, runs function $\text{KeyGen}(l)$: generate parameters $\langle G, G_T, e, q_1, q_2, g, h, N = q_1 \cdot q_2 \rangle$, q_1, q_2 are prime numbers of l bits, g, u is a ring generator of G , G is a ring of order N . Calculate secrets parameters $SB = g^{q_1}$, $PB = e(g, g)^{q_1}$; $h = u^{q_1}$. The TA selects random parameters as the private key sk_{TA} , then calculates the public key $pk_{TA} = g^{sk_{TA}}$. The TA selects an asymmetric encryption algorithm $E()$, hash function $H_1: \{0, 1\}^* \rightarrow G$, hash function $H_2: \{0, 1\}^* \rightarrow Z^*_{q_2}$. Finally, the TA keeps secret parameters $\langle q_1, sk_{TA} \rangle$; public parameters $\langle G, G_T, e, g, h, N, PK_S, E(), H_1(), H_2() \rangle$.

Phase 2: Registration of OAS: The OAS needs to register with the TA to obtain the qualification to apply for the fingerprint template. When registering in the TA, the OAS must select a random number as its private key $sk_S \in Z^*_N$, and calculates the public key $pk_S = g^{sk_S}$, then submits the public key and the information of the OAS to the TA, then the TA will assign the OAS a secret verification code IC_S and OAS ID number ID_S .

Phase 3: Registration of User (U): the user selects a random number as his private key $sk_{U_i} \in Z^*_N$ and calculates the public key $pk_{U_i} = g^{sk_{U_i}}$. The public key information is sent to the TA. In this case, The TA selects two random masking parameters $k_i, k'_i \in Z_N$ and the user ID number ID_{U_i} for each user,

TABLE 2
Time to Obtain $SB^{2H_2(k_i+c_s)}$ in Different Dimensions

Dimension n	100	200	300	400	500	600	700	800	900	1000
Time/ms	43483.3	48140.5	49116.3	49211.2	50098.1	49989.9	49937.2	49969.5	49931.1	49894.0

and calculates $IC_{SU_i} = H_2(IC_S, ID_{U_i}, k_i)$ and the user's secret parameters $SB_{U_i} = g^{q_1 \cdot k_i'}$, $PB_{U_i} = e(g, g)^{q_1 \cdot k_i'}$. The TA sends the parameters $\{SB_{U_i}, PB_{U_i}, IC_{SU_i}\}$ to U.

Part B: Fingerprint Template Generation

After the end of user registration, the TA collects the registered user's fingerprint and gets the FingerCode vector $\mathbf{x}_{U_i} = (x_1, x_2, \dots, x_n)$ of the user's fingerprint through image processing and feature extraction, where each x_i is an 8-bit data. For the FingerCode vector of each user's fingerprint template, the TA encrypts and stores the data through the following steps:

Phase 1: Data masking processing: use the user's masking parameters k_i and hash function $H_2()$ mask fingerprint characteristics, where $x'_j = x_j + IC_{SU_i}$, $IC_{SU_i} = H_2(IC_S, ID_{U_i}, k_i)$ $j \in \{1, \dots, n\}$. IC_S are the service provider identification codes.

Phase 2: Template generation and Bloom filter generation: (1) Artificially add a new fingerprint feature point $x'_{n+1} = 1$ to the fingerprint vector $\mathbf{x}_{U_i} = (x_1, x_2, \dots, x_n)$, and the TA selects $n+1$ random numbers to generate fingerprint template data $F_{U_i} = (f_{x_1}, f_{x_2}, \dots, f_{x_{n+1}}, f'_x)$. The specific calculation method is shown in Formula (2).

$$F_{U_i} = \begin{cases} f_{x_1} = g^{x'_1} \cdot h^{r_1} \\ f_{x_2} = g^{x'_2} \cdot h^{r_2} \\ \vdots \\ f_{x_n} = g^{x'_n} \cdot h^{r_n} \\ f_{x_{n+1}} = g^{x'_{n+1}} \cdot h^{r_{n+1}} \\ f'_x = PB_{U_i}^{(x'_1)^2 + (x'_2)^2 + \dots + (x'_n)^2} \end{cases} \quad (2)$$

IC_{SU_i} can ensure that the encrypted fingerprint templates F_{U_i} produced by the same user for different OAS are very different, and different users use different PB_{U_i} , which can prevent corrupt users and OAS from conducting collusion attacks.

(2) The TA calculates a set of data RDS_i for each user U_i , where $RDS_i = \{RD_{i,1}, RD_{i,2}, \dots, RD_{i,\Delta_d^2}\}$, $RD_{i,j} = PB_{U_i}^{j}$, $j \in \{0, 1, \dots, \Delta_d^2\}$. Δ_d is the Euclidean distance threshold for judging whether two FingerCodes match. Based on RDS_i , TA constructs a Bloom filter BF_{RDS_i} for each user.

Phase 1: User temporary fingerprint disturbance parameter generation: The TA generates l LWE samples $\mathbf{t}^{(1)}, \dots, \mathbf{t}^{(l)}$ for each user, among which $\mathbf{t}^{(j)} \in \mathbb{Z}_N^{n+1}$. Randomly selects $t_i^{(j)}$, $i \in [n]$, calculates $\mathbf{t}_{n+1}^{(j)} = \sum_{i=1}^n -t_i^{(j)} x'_i + e_j \bmod N$. n is the dimension of the feature vector, where e_j is obeys the discrete Gaussian distribution and x'_i is the masking data. TA returns $\mathbf{t}^{(1)}, \dots, \mathbf{t}^{(l)}$ to the user. It can get e_j .

$$\begin{aligned} \sum_{i=1}^{n+1} t_i^{(j)} x'_i &= \sum_{i=1}^n t_i^{(j)} x'_i + t_{n+1}^{(j)} \cdot x'_{n+1} \\ &= \sum_{i=1}^n t_i^{(j)} x'_i + \sum_{i=1}^n -t_i^{(j)} x'_i + e_j \\ &= e_j \end{aligned}$$

4.2 Encrypted Template Authorization

If the user wants to use the service provided by the OAS, the user must register in the OAS.

Phase 1: Users register in OAS: The user signs his ID with his private key and sends it to OAS, $\langle ID_S || ID_{U_i} || TS_1 || Sig_{U_i} \rangle$, where $Sig_{U_i} = H_1(ID_S || ID_{U_i} || pk_S || TS_1)^{sk_{U_i}}$, TS_1 is the timestamp and ID_{U_i} is the ID number of each user. After OAS receives the user's request data, it first verifies the timeliness of TS_1 and then verifies whether the equation $e(g, Sig_{U_i}) = e(PK_{U_i}, H_1(ID_S || ID_{U_i} || pk_S || TS_1))$ holds. If the equation holds, the verification passes.

Phase 2: OAS applies for user template: After the OAS verifies the user's application data, it signs $Sig_S = H_1(ID_S || TS_2)^{sk_S}$ with its own private key and sends $\langle ID_{U_i} || TS_1 || Sig_{U_i} || ID_S || TS_2 || Sig_S \rangle$ to the TA.

Phase 3: The TA provides a fingerprint template: when the TA receives the OAS's request, the TA first verifies the timeliness of the signed timestamps TS_1 and TS_2 , as well as the user identification U_i and OAS ID_S , and then uses the user's public key PK_{U_i} and OAS's PK_S to verify the correctness of the signature, that is, verify that the equations $e(g, Sig_{U_i}) = e(PK_{U_i}, H_1(ID_S || ID_{U_i} || pk_S || TS_1))$ and $e(g, Sig_S) = e(pk_S, H_1(ID_S || TS_2))$ are correct. Sex. If these two equations are both true, the signature of the user and the fingerprint authentication server is valid, and the TA passes the fingerprint template request of the OAS. TA uses the private key SK to construct a signature SIG. After the signature is completed, the TA sends the fingerprint template data package $\langle ID_S || ID_{U_i} || F_{U_i} || BF_{RDS_i} || TS_3 || Sig_{TA} \rangle$ to the OAS.

Phase 4: OAS saves the user's template information: After receiving the fingerprint template data packet from TA, OAS first verifies the timeliness of the time stamp information TS_3 and then verifies the TA's signature information $e(g, Sig_{TA}) = e(pk_{TA}, H_1(ID_S || ID_{U_i} || F_{U_i} || BF_{RDS_i} || TS_3))$. If there is no problem with the signature information, the OAS saves the user's authorized fingerprint template data package $\langle ID_{U_i} || F_{U_i} || BF_{RDS_i} \rangle$.

4.3 Authentication Query Generation

After the user registers at TA and OAS, it can make a service request to the corresponding OAS.

Phase 1: Fingerprint information collection: The user uses the terminal device to collect his own fingerprint image, and obtains the vector information $Y_{U_i} = (y_1, y_2, \dots, y_n)$ of the fingerprint through the image processing and feature extraction of the Gabor filter, and then uses the IC_{SU_i} to mask the vector information to generate the mask data $y'_j = y_j + IC_{SU_i}$, where $j \in \{1, \dots, n\}$.

Phase 2: Generate scrambling ciphertext offset t : The user randomly selects $s \in \{0, 1\}^l$, and calculates the scrambling ciphertext offset t , where $t = \langle (t^{(1)}, \dots, t^{(l)}), s \rangle$, $t^{(1)}, \dots, t^{(l)}$ are the scrambling parameters are given to the user by the trusted authority center.

Phase 3: User temporary fingerprint ciphertext construction: The user uses the fingerprint matching threshold Δ_d and the scrambled ciphertext offset t to construct the fingerprint information $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_{n+1}}, rq'_{y_j})$ for verification by the user. The specific calculation formula is shown in (3).

$$RQ_{U_i} = \begin{cases} rq_{y_1} = SB_{U_i}^{2y'_1+t_1} \\ \vdots \\ rq_{y_n} = SB_{U_i}^{2y'_n+t_n} \\ rq_{y_{n+1}} = SB_{U_i}^{t_{n+1}} \\ rq'_{y_j} = PB_{U_i}^{y'_1{}^2+y'_2{}^2+\dots+y'_n{}^2-\Delta_d^2} \end{cases} \quad (3)$$

It is worth noting that for different OAS, the generated temporary fingerprint ciphertext RQ_{U_i} is different. At the same time, because of the disturbance bias t , the temporary fingerprint ciphertext RQ_{U_i} generated each time when the user makes a verification request is also different. This effectively prevents collusion attacks between OAS and exhaustive attacks by OAS.

Phase 4: Submission of temporary fingerprint ciphertext: The user uses the private key sk_{U_i} to sign the temporary fingerprint ciphertext RQ_{U_i} . The signature information is $Sig_{U_i} = H_1(RQ_{U_i} || ID_{U_i} || TS_4)^{sk_{U_i}}$, and the user sends a verification request packet $\langle RQ_{U_i} || ID_{U_i} || TS_4 || Sig_{U_i} \rangle$ to the OAS after the signature is completed.

4.4 Fingerprint Matching on Cloud Server

After receiving the user authentication request, OAS uses the fingerprint template to perform authentication services.

Phase 1: Verify user service request packet: First, after OAS receives the user's verification request packet $\langle RQ_{U_i} || ID_{U_i} || TS_4 || Sig_{U_i} \rangle$, it first verifies the user's identity ID_{U_i} and the timeliness of the time stamp TS_4 , and then uses the user's public key PK_{U_i} to verify the user's signature Sig_{U_i} .

$e(g, Sig_{U_i}) = e(PK_{U_i}, H_1(RQ_{U_i} || ID_{U_i} || TS_4))$, If the equation holds, the user's signature is valid, and the user's request is successful.

Calculating matching parameters: According to the user's ID_{U_i} , OAS finds the matching template $\langle ID_{U_i} || F_{U_i} || BF_{RDS_i} \rangle$ corresponding to the user in the database and obtains the user's encrypted fingerprint template data F_{U_i} and bloom filter BF_{RDS_i} . OAS uses the fingerprint template data and the user's temporary fingerprint ciphertext to calculate the matching parameter M_d . The specific formula is shown in (4). The service provider runs BF_Test on the Bloom filter BF_{RDS_i} to determine whether the matching parameter M_d belongs to the reference set RDS_i , and obtains the matching result RS .

$$\begin{aligned} RQ_{U_i} &= (rq_{y_1}, rq_{y_2}, \dots, rq_{y_{n+1}}, rq'_{y_j}), F_{U_i} = (f_{x_1}, f_{x_2}, \dots, f_{x_{n+1}}, f'_x) \\ e(f_{x_j}, rq_{y_j}) &= e(g^{x'_j} \cdot h^{r_j}, g^{q_1 \cdot k'_i (2y'_j + t_j)}) \\ &= e(g^{x'_j} \cdot h^{r_j}, g^{q_1 \cdot k'_i (2y'_j + t_j)}) \\ &= e(g, g)^{x'_j q_1 \cdot k'_i (2y'_j + t_j)} \\ &= PB_{U_i}^{x'_j (2y'_j + t_j)} \\ &= PB_{U_i}^{2x'_j y'_j} \cdot PB_{U_i}^{x'_j t_j} \\ f'_x \cdot rq'_{y_j} &= PB_{U_i}^{(x'_1{}^2 + x'_2{}^2 + \dots + x'_n{}^2)} \cdot PB_{U_i}^{y'_1{}^2 + y'_2{}^2 + \dots + y'_n{}^2 - \Delta_d^2} \\ &= PB_{U_i}^{(x'_1{}^2 + \dots + x'_n{}^2 + y'_1{}^2 + \dots + y'_n{}^2 - \Delta_d^2)} \\ M_d &= \frac{\prod_{j=1}^{n+1} e(f_{x_j}, rq_{y_j})}{f'_x \cdot rq'_{y_j}} \\ &= \frac{\prod_{j=1}^n e(g^{x'_j} \cdot h^{r_j}, g^{q_1 \cdot k'_i (2y'_j + t_j)}) \cdot e(g^{x'_{n+1}} \cdot h^{r_{n+1}}, g^{q_1 \cdot k'_i t_{n+1}})}{f'_x \cdot rq'_{y_j}} \\ &= \frac{\prod_{j=1}^n e(g, g)^{x'_j q_1 \cdot k'_i (2y'_j + t_j)} \cdot e(g, g)^{x'_{n+1} q_1 \cdot k'_i t_{n+1}}}{e(g, g)^{q_1 \cdot k'_i \times (x'_1{}^2 + x'_2{}^2 + \dots + x'_n{}^2)} \cdot e(g, g)^{q_1 \cdot k'_i \times (y'_1{}^2 + y'_2{}^2 + \dots + y'_n{}^2 - \Delta_d^2)}} \\ &= \frac{e(g, g)^{q_1 \cdot k'_i \{2x'_1 y'_1 + \dots + 2x'_n y'_n + \sum_{j=1}^{n+1} t_j x'_j\}}}{PB_{U_i}^{(x'_1{}^2 + \dots + x'_n{}^2 + y'_1{}^2 + \dots + y'_n{}^2 - \Delta_d^2)}} \\ &= PB_{U_i}^{\Delta_d^2 - ((x_1 - y_1)^2 + \dots + (x_n - y_n)^2) + e'} \end{aligned} \quad (4)$$

Phase 2: Return the ciphertext of the matching result

- (1) OAS encrypted matching result $C_{RS} = E_{PK_{U_i}}(RS)$;
- (2) Send the matching result ciphertext to the user: $\langle E_{PK_{U_i}}(RS) || TS_5 || Sig_R \rangle$, where $Sig_R = H_1(E_{PK_{U_i}}(RS) || TS_5)^{sk_S}$.

Phase 3: The user decrypts the matching result

After the user verifies the validity of the signature and timestamp, it verifies whether the equation $e(g, sig_R) = e(pk_S, H_1(E_{PK_{U_i}}(RS) || TS_5))$ is established. If the verification passes, the user gets the matching result $RS = D_{SK_{U_i}}(C_{RS})$. If the result RS is true, the identity authentication is successful; otherwise, the identity authentication fails.

5 EXPERIMENT AND ANALYSIS

This section mainly analyzes the security and performance evaluation of the Secure e-finger scheme, and analyzes and compares the computational complexity, communication and storage costs of the e-Finga and Secure e-finger scheme.

5.1 Security Analysis

We analyze the security of the Secure e-finger solution from three aspects: user data privacy, data confidentiality, and message authentication.

(1) Privacy of user's temporary fingerprint

In this scheme, the temporary fingerprint features extracted and encrypted by the user are sent to the server provider. During the data transmission process, the attacker may monitor the communication channel and obtain the communication data, and the attacker may also be a semi-trusted OAS. For such attackers, this scheme can guarantee the privacy of user's temporary fingerprints. To prevent attackers exhaustive fingerprint feature vectors, we add the redundant $IC_{SU_i} = H_2(IC_S, ID_{U_i}, k_i)$ values to the user's feature vector $Y_{U_i} = (y_1, y_2, \dots, y_n)$ (this redundancy is secretly generated and stored by TA using the user I and the IC_S information of the corresponding OAS) to make the sample space extended, generates masking data $y'_j = y_j + IC_{SU_i}$, which can effectively protect the user's fingerprint characteristic data information. To prevent multiple temporary fingerprint ciphertext comparison attacks, each component $rq_{y_j} = SB_{U_i}^{2y'_j + t_j}$ in the temporary fingerprint ciphertext data $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_n}, rq'_{y_n})$ generated by the user i with a different ciphertext offset t , so that when the user makes a verification request, The temporary fingerprint ciphertext generated each time is independent of RQ_{U_i} . This method can also effectively prevent collusion attacks between OAS. To avoid attacks of collusion between OAS and corrupt users, we use different parameter pairs $\langle SB_{U_i}, PB_{U_i} \rangle$ for different users. Even if the attacker obtains the user's parameter pair information, the security of other user's information can be guaranteed. Even if the attacker obtains the parameters and wants to calculate the user fingerprint data $Y_{U_i} = (y_1, y_2, \dots, y_n)$ is a discrete logarithm problem. Such a discrete logarithm problem is difficult in a finite field. In summary, this solution can effectively ensure the privacy of user fingerprint characteristic data.

(2) Confidentiality of template data

In this scheme, the encrypted template data will be authorized to the OAS. In the process of data transmission, the attacker may eavesdrop on the communication channel and obtain communication data, and the attacker may also be a semi-trusted OAS. For such attackers, this scheme can guarantee the confidentiality of the user's template data.

First, To prevent attackers from attacking the fingerprint template by exhaustively exhausting fingerprint feature vectors, the trusted authority center adds a secret redundant value IC_{SU_i} to the FingerCode vector $x_{U_i} = (x_1, x_2, \dots, x_n)$ of the fingerprint so that the sample space is expanded to generate masking data $x'_j = x_j + IC_{SU_i}$, and further use random The number r_j is encrypted to obtain the fingerprint template data $F_{U_i} = (f_{x_1}, f_{x_2}, \dots, f_{x_{n+1}}, f'_n)$, so that the encrypted template values of the same feature components are

irrelevant. Second, the fingerprint feature is added to the redundancy and contains the OAS's identification code IC_{SU_i} . Hence, the template data obtained by each OAS is different, which can effectively prevent the abuse of user template data. Finally, for different users, different reference data sets $RDS_i = \{RD_{i,1}, RD_{i,2}, \dots, RD_{i,\Delta_d^d}\}$ are used to construct a user-specific Bloom filter BF_{RDS_i} . The OAS cannot obtain the specific data of the reference data set through the Bloom filter. The OAS cannot get any information about the template data except the final matching result. Therefore, this solution can effectively guarantee the confidentiality of fingerprint template data.

(3) Security of Information Communication

All information interactions between the U, OAS, and TA contain digital signature information. All three parties use the BLS short signature technology to sign. The BLS short signature is provably safe in the random oracle based on the CDH problem [23], and can achieve effective message authentication. By verifying the timeliness, integrity and source of the message, it is determined whether the communication message has been changed or replaced by the attacker, and the validity of the message is verified.

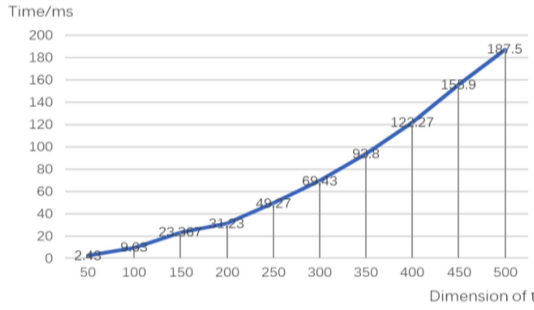
While ensuring the timeliness, integrity, and source of the message, the malicious OAS attacker C may also steal data packets in the user and normal OAS communication channels, and use the data packets to request a user template from the trusted authority center. In the solution in this article, when the user registers with the authentication provider, he adds the ID information of the OAS to the data packet. The data packet is $\langle ID_S || ID_{U_i} || TS_1 || Sig_{U_i} \rangle$, where $Sig_{U_i} = H_1(ID_S || ID_{U_i} || PK_S || TS_1)^{sk_{U_i}}$. Even if there is a malicious OAS attacker C in the communication channel, because the data packet contains the user information ID_{U_i} and the normal OAS information ID_S , the malicious OAS attacker C cannot use the data packet to request users from the trusted authority center. The template information can effectively prevent the template data from leaking to the unauthorized OAS.

5.2 Secure E-Finger Scheme Performance Evaluation

In the Secure e-finger scheme, the server performs Euclidean distance matching on ciphertext data. The calculation process is the same, and the time and space costs are the same as the e-Finga solution.

The running time of the TA terminal does not affect the user's service request experience, only the user's registration time in the TA. In the Secure e-finger scheme, the TA needs to generate a disturbing ciphertext offset t for each user, so the time has increased in the user's registration at the TA. The system initialization phase of the Secure e-finger solution only affects the user registration time. It does not affect the efficiency of the user end and the efficiency of the server provider. As shown in Fig. 4, the time required for the TA to generate different dimensions of the scrambled ciphertext offset t , It takes no more than 190 ms to generate 500-dimensional t .

In the Secure e-finger scheme, the client encrypts the FingerCode vector of the user's fingerprint and calculates the temporary fingerprint ciphertext $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots,$

Fig. 4. Times when TA generates t .

rq_{y_n}, rq'_y). The most important parameters that affect the calculation efficiency are the vector dimension n and the increased scrambling ciphertext offset t . This experiment selects fingerprint feature vectors of different dimensions to test the computing performance of the client. The vector dimension selects ten sets of data from 100 to 1000. The experimental results show that the calculation time increases linearly, as shown in Fig. 5 and Table 3. The calculation cost of the client of the Secure e-finger solution has increased by 6.31 percent in the 100 dimensions compared with the original solution. In other dimensions, the calculation cost has only increased by 3% percent and can fully meet the needs of actual application scenarios.

5.3 Computational Complexity and Communication and Storage Costs

The proposed Secure e-finger scheme can offer an online efficient fingerprint authentication service. We assume the FingerCode is n -dimensional feature vector. When TA generates the encrypted template $F_{U_i} = (f_{x_1}, f_{x_2}, \dots, f_{x_{n+1}}, f'_n)$, it requires $2(n+1) + 1$ exponentiation operations and $2(n+1)$ multiplication operations. When the user generates the encrypted fingerprint information $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_{n+1}}, rq'_y)$ in the Query Generation phase, it requires $n+2$ exponentiation operations and $2n$ multiplication operations. In the fingerprint matching phase, it will cost OASer $n+1$ pairing operations and $n+2$ multiplication operations. Denote the computational costs of an exponentiation operation, a multiplication operation, and a pairing operation by C_e , C_m , C_p , respectively. Then, totally for TA, the user, and OASer, the computational cost will be $(2(n+1) + 1) \cdot C_e + 2(n+1) \cdot C_m$, $(n+2) \cdot C_e + 2n \cdot C_m$ and $(n+1) \cdot C_p + (n+2) \cdot C_m$ in the Secure e-finger.

The proposed Secure e-finger uses lightweight multi-party random masking, LWE samples, and polynomial aggregation techniques, which can provide efficient online fingerprint authentication while preserves the privacy of the fingerprint information with low overhead in computation. In the following, we compare with the e-Finga scheme.

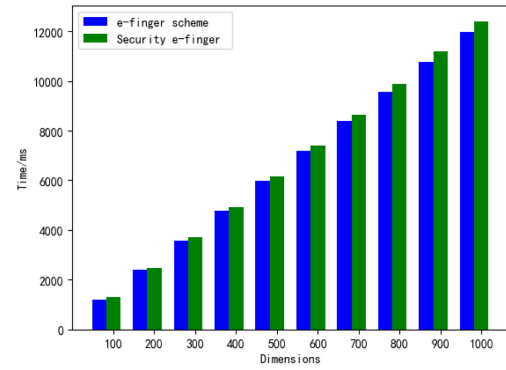


Fig. 5. Comparison of the calculation cost of the two schemes.

TABLE 4
Comparison of Computation Complexity

	User	Server
Secure e-finger	$(n+2) \cdot C_e + 2n \cdot C_m$	$(n+1) \cdot C_p + (n+2) \cdot C_m$
e-Finga	$(n+1) \cdot C_e + 2n \cdot C_m$	$n \cdot C_p + (n+1) \cdot C_m$
Increase rate / %	$< \frac{1}{n+1}$	$< \frac{1}{n}$

The corresponding computational costs of the user and the server in e-finger are $(2n+1) \cdot C_e + 2n \cdot C_m$, $(n+1) \cdot C_e + 2n \cdot C_m$ and $n \cdot C_p + (n+1) \cdot C_m$ respectively. We present the computation complexity comparison of e-Finga and Secure e-finger in Table 4. It can be concluded from the table that the computational complexity of the client-side of the Secure e-finger scheme is C_e more than that of the e-Finga scheme, and the computational complexity of the server-side is $C_p + C_m$ more than that of the e-Finga scheme. The increase in computational complexity is less than $\frac{1}{n}$. Usually n is very large. In actual operation, the increase in computational complexity is less than 1% percent.

In the two schemes, the difference in user's query is that $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_{n+1}}, rq'_y)$ in the Secure e-finger scheme and $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_n}, rq'_y)$ in the e-Finga scheme.

We have made a comparison of storage place costs between Secure e-finger and e-Finga. In the Secure e-finger scheme, the user side needs to store the user's l LWE samples $t^{(1)}, \dots, t^{(l)}$ (the value $l = n$ here). The server needs to store the $\langle ID_{U_i} \| F_{U_i} \| BF_{RDSi} \rangle$ of each user, while the e-Finga scheme only needs to store each user's $\langle ID_{U_i} \| F_{U_i} \rangle$. The value Δ_d is 5% percent here. The number of users is N . We calculate the space of the storage place of the above two schemes, and the results are shown in Table 5. Considering that the server has a strong storage capacity, the increase in

TABLE 3
Comparing the Calculation Time of the Two Schemes

Dimension n	100	200	300	400	500	600	700	800	900	1000
e-Finga/ms	1203.89	2394.7	3579.6	4770.7	5991.9	7168.4	8377.6	9571.9	10760.7	11975.7
Secure e-finger /ms	1279.8	2471.2	3697.4	4929	6152.4	7412.8	8638.6	9862.9	11174	12406.4
Increase rate / %	6.31	3.19	3.29	3.32	2.68	3.41	3.12	3.04	3.84	3.60

TABLE 5
Comparison of Storage Space Cost (n = 640)

	User	Server
Secure e-finger	16B	(20 KB+3.125 KB)*N
e-Finga	0	20 KB*N

data volume will not cause too much burden on the server. Following the common optimization techniques in lattice cryptography, one may simply store a small seed for the l LWE samples $t^{(1)}, \dots, t^{(l)}$ to minimize storage. We use a 128-bit seed to generate l LWE samples $t^{(1)}, \dots, t^{(l)}$, following one of the submissions of NIST PQC, FrodoKEM [24]. The storage space cost in user increased by 16B, increased by 15.625% percent on the server-side.

In addition, we have made a comparison of communication costs between Secure e-finger and e-Finga. In e-Finga, the user's query is $\langle RQ_{U_i} || ID_{U_i} || TS_4 || Sig_{U_i} \rangle$, and the response is the authentication result RS , where RS is a boolean value. We calculate the size of the query package and response package of the above two schemes, and the results are shown in Table 6. The communication cost on the server side remains unchanged, and the communication cost on the user side only increased by 0.3125% percent.

6 THRESHOLD AUTHENTICATION SCHEME BASED ON BIOMETRICS

Threshold cryptography provides a way in which the secret key of an organization or a company can be shared practically. For the protection of important targets, which are of high value and need key protection such as important safes, partnership company accounts, single user authentication is likely to cause excessive authority problems. Single user identity authentication can completely control important targets. This user has ownership of important goals, which makes the user's authority over important goals too large. If the user is dishonest and credible, it will cause great harm. The usual practice is that we divide the key into several parts, and each user has a part of the key. In response to such problems, based on the Secure e-finger scheme, we propose a threshold scheme framework based on biological characteristics. Threshold identity authentication is a special identity authentication. The biometric-based secret threshold authentication scheme consists of three phases, the secret distribution phase, the secret reconstruction phase, and the identity authentication phase. In the secret distribution phase, TA divides the master key into multiple shares. In the secret reconstruction stage, using the multiplicative homomorphism of power operation, the client reconstructs

TABLE 6
Comparison of Communication Costs (n = 640)

	User	Server
Secure e-finger	20.0625KB	1B
e-Finga	20KB	1B
Increase rate / %	0.3125	0

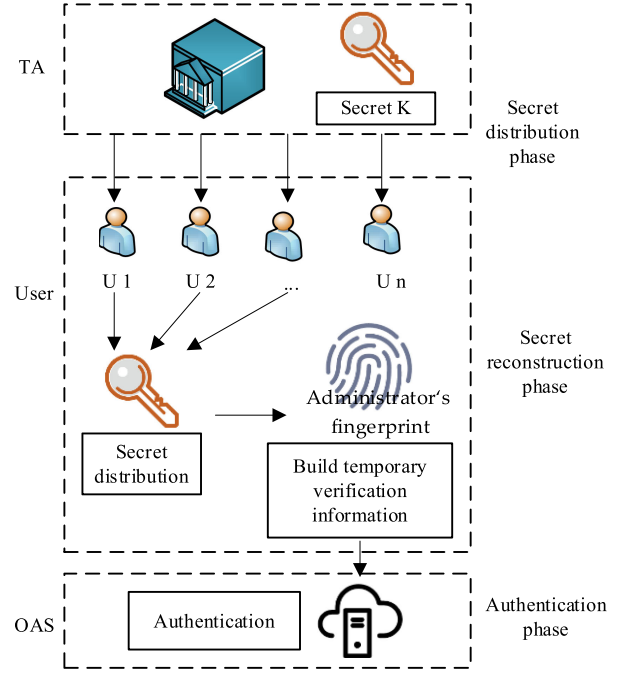


Fig. 6. System structure diagram.

the secret value on the power. The system structure diagram is shown in Fig. 6.

1) Secret distribution phase

TA initialization: TA selects a security parameter $l \in Z^+(l > 512)$, runs function $\text{KeyGen}(l)$: generate parameters $\langle G, G_T, e, q_1, q_2, g, h, q_1, q_2, p \rangle$, q_1, q_2, p are prime numbers of l bits, g, u is a ring generator of G , G is a ring of order $N = q_1 \cdot q_2$. Randomly select $K \in Z_{q_1}$ as the secret value of the threshold scheme. Calculate secrets parameters $SB_s = g^{q_1 \cdot K}$, $PB_s = e(g, g)^{q_1 \cdot K}$, $h = u^{q_1}$. The TA selects random parameters as the private key sk_{TA} , then calculates the public key $pk_{TA} = g^{sk_{TA}}$. The TA selects an asymmetric encryption algorithm $E()$, hash function $H_1: \{0, 1\}^* \rightarrow G$, hash function $H_2: \{0, 1\}^* \rightarrow Z_{q_2}^*$. Finally, the TA keeps secret parameters $\langle q_1, sk_{TA} \rangle$; public parameters $\langle G, G_T, e, g, h, N, PK_S, E(), H_1(), H_2() \rangle$. TA initializes the number n' of secret sharing shares and threshold k of the threshold scheme. TA randomly selects the coefficient $a_1, a_2, \dots, a_{k-1} \in Z_{q_1}$ of secret sharing.

Registration of User (U): the user $U_1, U_2, \dots, U_{n'}$ select random number as his private key $sk_{U_i} \in Z_N^*$ and calculates the public key $pk_{U_i} = g^{sk_{U_i}}$. The public key information is sent to the TA. TA divides the secret value $SB_s = g^{q_1 \cdot K}$ into shares $s(x)$, where $x = \{1, 2, \dots, n'\}$, and distributes it to users $U_1, U_2, \dots, U_{n'}$. The secret division method is shown in formula (5).

$$s(x) = g^{q_1 \cdot K} * g^{\sum_{i=1}^{k-1} a_i x^i} \mod p, \{x = 1, 2, \dots, n'\} \quad (5)$$

TA calculates $IC_{SU_i} = H_2(IC_S, ID_{U_i}, s(i))$ and the user's secret parameters $SB_{U_i} = s(i)PB_{U_i} = e(g, s(i))$. The TA sends the parameters $\{SB_{U_i}, PB_{U_i}, IC_{SU_i}\}$ to each U_i . TA chooses a user as the administrator, and uses the secret value $SB_s = g^{q_1 \cdot K}$ to encrypts the user's fingerprint information to generate template data $F_{U_i} = (f_{x_1}, f_{x_2}, \dots, f_{x_{n+1}}, f'_n)$. See Section 4.1 for the specific generation method.

2) Secret reconstruction phase

TABLE 7
Comparison of Computation Complexity.

	User	Server
Secure e-finger	$(n+2) \cdot C_e + 2n \cdot C_m$	$(n+1) \cdot C_p + (n+2) \cdot C_m$
Threshold scheme	$(n+2+k \cdot \prod_{l=1, l \neq j}^k \frac{i_l}{i_j - i_l}) \cdot C_e + (2n+k) \cdot C_m$	$(n+1) \cdot C_p + (n+2) \cdot C_m$

When the secret needs to be reconstructed, k shares are needed, that is, the secret value $g^{q_1 \cdot K}$ can be reconstructed using $\{(i_j, s(i_j)) | j = 1, 2, \dots, k\}$. Reconstruction method as shown in Formula 6.

$$g^{q_1 \cdot K} = (-1)^{k-1} \prod_{j=1}^k \left(s(i_j) \prod_{l=1, l \neq j}^k \frac{i_l}{i_j - i_l} \right) \mod p \quad (6)$$

3) The identity authentication phase

After the secret reconstruction is completed, the client uses the secret $g^{q_1 \cdot K}$ to encrypt the fingerprint information of the admin user to generate temporary verification information $RQ_{U_i} = (rq_{y_1}, rq_{y_2}, \dots, rq_{y_{n+1}}, rq_y)$. See Section 4.3 for detail. Then send RQ_{U_i} to the OAS for identity authentication.

In terms of the security of the scheme, the threshold scheme uses the Secure e-finger scheme to protect the user's privacy, and the template data for identity authentication is stored in the TA. The attack obtains the template data F_{U_i} and RQ_{U_i} verification request data in the channel. The Secure e-finger scheme ensures that the attack cannot use these two data to recover the user's fingerprint characteristics $\mathbf{x}_{U_i} = (x_1, x_2, \dots, x_n)$, shared secret value $g^{q_1 \cdot K}$, and reconstruct the verification request data RQ'_{U_i} . The threshold in the threshold scheme uses Shamir threshold scheme, and $s(x)$ is the user's private key, which is also the share in the threshold scheme. In the threshold scheme, only when k shares are obtained can the client reconstruct the secret value $g^{q_1 \cdot K}$. The client can reconstruct the secret value to perform identity authentication. The security of this scheme is based on the Shamir threshold scheme and the security of the Secure e-finger scheme. The shares are set up and distributed by the security authority TA to ensure the safety and reliability of the secret distribution. Only the admin user, or the number of shares does not meet k , cannot perform identity authentication to ensure absolute privacy security.

In terms of the efficiency of the scheme, denote the computational costs of an exponentiation operation, a multiplication operation, and a pairing operation by C_e , C_m , C_p , respectively. Then, totally for TA, the user, and OAS er, the computational cost will be $(2n+n' \sum_{i=1}^{k-1} a_i x^i + 3) \cdot C_e + (2n+n'+2) \cdot C_m$, $(n+2+k \cdot \prod_{l=1, l \neq j}^k \frac{i_l}{i_j - i_l}) \cdot C_e + (2n+k) \cdot C_m$ and $(n+1) \cdot C_p + (n+2) \cdot C_m$ in the threshold scheme. In practical applications, the values of k and n' are generally small. We present the computation complexity comparison of Threshold scheme and Secure e-finger in Table 7.

7 CONCLUSION

This article discusses and analyzes the security of user fingerprint data in the e-Finga scheme and finds that there are three hidden dangers in the e-Finga scheme: temporary fingerprints use deterministic encryption algorithms, multiple users use the same secret system parameters, and the authorization data package does not authenticate the identity of the authorized object in the Encrypted Template Authorization phase. Aiming at these hidden dangers, this paper proposes a temporary fingerprint ciphertext attack method and demonstrates the feasibility of the attack method through experiments. The experimental results show that the attacker can obtain the user's fingerprint feature plaintext information when the user's temporary fingerprint ciphertext is known.

At the same time, this paper proposes a new security and privacy protection scheme for the flaws in the e-Finga scheme, the Secure e-finger, which introduces samples of difficult problems on the lattice to protect the privacy of user's temporary fingerprint characteristics. The Secure e-finger can solve the security problem of the privacy protection of fingerprint data in the online fingerprint authentication system while considering the system's requirements for efficiency, providing a new privacy protection technical solution for the biometric authentication system. In order to meet the needs in actual application scenarios, we propose a threshold scheme framework based on the Secure e-finger scheme.

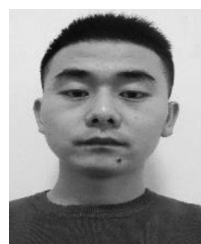
ACKNOWLEDGMENT

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0802000, in part by the National Natural Science Foundation of China under Grants 61872384, 62172436, and 62102452, in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2020JQ-492, in part by the Innovative Research Team in Engineering University of PAP under Grant KYTD201805, and in part by the Fundamental Research Project of Engineering University of PAP under Grant WJY201912, Tanping Zhou and Yao Liu contributes equally to this work.

REFERENCES

- [1] G. Chen, F. Wang, X. Yuan, Z. Li, Z. Liang, and A. Knoll, "NeuroBiometric: An eye blink based biometric authentication system using an event-based neuromorphic vision sensor," *IEEE/CAA J. Automat. Sinica*, vol. 8, no. 1, pp. 206–218, Jan. 2021.
- [2] R. Van Noorden, "The ethical questions that haunt facial-recognition research," *Nature*, vol. 587, no. 7834, pp. 354–358, Nov. 2020.
- [3] S. Shin, J. Jung, M. Kang, K. H. Kang, and Y. T. Kim, "Electromyogram-based algorithm using bagged trees for biometric person authentication and motion recognition," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2020, pp. 1–4.
- [4] C. S. Network, "Biometric database leads to data leakage of millions of users," Informatization Softw. Service Netw.-Helping Construction Digit. China (infosws.cn), 2019. [Online]. Available: <http://security.infosws.cn/20190815/25694.htm>

- [5] Z. Rui and Z. Yan, "A survey on biometric authentication: Toward secure and privacy-preserving identification," *IEEE Access*, vol. 7, pp. 5994–6009, 2019.
- [6] R. Miller, N. K. Banerjee, and S. Banerjee, "Within-system and cross-system behavior-based biometric authentication in virtual reality," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces Abstr. Workshops*, 2020, pp. 311–316.
- [7] A. Bodo, "Method for producing a digital signature with aid of a biometric feature (1994)," *German Patent*, vol. 42, no. 43, 1994.
- [8] M. Barni, G. Droandi, R. Lazzeretti, and T. J. I. B. Pignata, "SEMBA: Secure multi-biometric authentication," *IET Biometrics*, vol. 8, pp. 411–421, 2019.
- [9] S. Aziz, M. U. Khan, Z. A. Choudhry, A. Aymin, and A. Usman, "ECG-based biometric authentication using empirical mode decomposition and support vector machines," in *Proc. IEEE 10th Annu. Inf. Technol., Electron. Mobile Commun. Conf.*, 2019, pp. 0906–0912.
- [10] K. Honda, M. Omori, S. Ubukata, and A. Notsu, "A study on fuzzy clustering-based k-anonymization for privacy preserving crowd movement analysis with face recognition," in *Proc. 7th Int. Conf. Soft Comput. Pattern Recognit.*, 2015, pp. 37–41.
- [11] A. T. B. Jin, D. N. C. Ling, and A. Goh, "Biobhashing: Two factor authentication featuring fingerprint data and tokenised random number," *Pattern Recognit.*, vol. 37, no. 11, pp. 2245–2255, 2004.
- [12] M. Elhoseny, E. Essa, A. Elkhateb, A. E. Hassanien, and A. Hamad, "Cascade multimodal biometric system using fingerprint and iris patterns," in *Proc. Int. Conf. Adv. Intell. Syst. Informat.*, 2017, pp. 590–599.
- [13] M. Blanton and P. J. B. S. Gasti, "Secure and efficient iris and fingerprint identification," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2015, pp. 190–209.
- [14] M. Steinbauer *et al.*, "Privacy-preserving biometrics authentication systems using fully homomorphic encryption," *Int. J. Pervasive Comput. Commun.*, 2015, pp. 190–209.
- [15] M. Haghighat, S. Zonouz, and M. J. E. S. W. A. Abdel-Mottaleb, "CloudID: Trustworthy cloud-based and cross-enterprise biometric identification," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7905–7916, 2015.
- [16] T. Hoang, A. A. Yavuz, and J. G. J. I. T. O. S. C. Merchan, "A secure searchable encryption framework for privacy-critical cloud storage services," *IEEE Trans. Serv. Comput.*, early access, Feb. 1, 2019, doi: [10.1109/TSC.2019.2897096](https://doi.org/10.1109/TSC.2019.2897096).
- [17] M. J. I. S. J. A. G. P. Yasuda, "Secure hamming distance computation for biometrics using ideal-lattice and ring-LWE homomorphic encryption," *Inf. Secur. J.: Glob. Perspective*, vol. 26, no. 2, pp. 85–103, 2017.
- [18] H. Zhu, Q. Wei, X. Yang, R. Lu, and H. Li, "Efficient and privacy-preserving online fingerprint authentication scheme over outsourced data," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 576–586, Apr.–Jun. 2021.
- [19] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography Conference*. Berlin, Germany: Springer, 2005, pp. 325–341.
- [20] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Proc. Int. Workshop Public Key Cryptogr.*, 2004, pp. 277–290.
- [21] O. Regev, "On lattices, learning with errors, random linear codes and cryptography," *J. ACM*, vol. 56, pp. 1–40, 2009.
- [22] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based fingerprint matching," *IEEE Trans. Image Process.*, vol. 9, no. 5, pp. 846–859, May 2000.
- [23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2003, pp. 416–432.
- [24] E. Alkim *et al.*, "FrodoKEM learning with errors key encapsulation," Submission to the NIST Post-Quantum Standardization project, 2017–2019. [Online]. Available: <https://frodokey.org/#code>



Yao Liu was born in Liaocheng, China, in 1993. He is currently working toward the master's degree in engineering from the University of People's Armed Police. His research interests include the areas of applied cryptography, cybersecurity, and privacy protection.



Tanping Zhou was born in Yingtan, China in 1989. He received the PhD degree in engineering from the University of People's Armed Police. His research interests include fully homomorphic encryption, encryption scheme based on lattice.



Zelun Yue was born in Shijiazhuang, China, in 1989. He received the master's degree from the University of People's Armed Police. His research interests include cryptography and information security.



Wenchao Liu was born in Wuwei, China, in 1994. He received the master's degree in engineering from the University of People's Armed Police. His research interests include fully homomorphic encryption and information security.



Yiliang Han was born in 1978. He received the PhD degree. He is currently a PhD supervisor. His research interests include information security and cryptology.



Qi Li was born in Suzhou, China, in 1994. He received the master's degree from Xidian University. He is currently working toward the PhD degree with the University of Guelph. He research interests include applied cryptography and AI security.



Xiaoyuan Yang was born in Xiangtan, China, in 1959. He is currently a PhD supervisor of engineering with the University of People's Armed Police. His research interests include information security and cryptology.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.