Hindawi
Security and Communication Networks
Volume 2020, Article ID 2523834, 14 pages
https://doi.org/10.1155/2020/2523834

Research Article

Efficient Hierarchical Authentication Protocol for Multiserver Architecture

Jiangheng Kou, Mingxing He, Ling Xiong, and Zeqiong Lv

School of Computer and Software Engineering, Xihua University, Chengdu, Sichuan 610039, China

Correspondence should be addressed to Mingxing He; he_mingxing64@aliyun.com

Received 23 August 2019; Revised 14 November 2019; Accepted 22 January 2020; Published 23 March 2020

Academic Editor: José María de Fuentes

The multiserver architecture authentication (MSAA) protocol plays a significant role in achieving secure communications between devices. In recent years, researchers proposed many new MSAA protocols to gain more functionality and security. However, in the existing studies, registered users can access to all registered service providers in the system without any limitation. To ensure that the system can restrict users that are at different levels and can access to different levels of service providers, we propose a new lightweight hierarchical authentication protocol for multiserver architecture using a Merkle tree to verify user's authentication right. The proposed protocol has hierarchical authentication functionality, high security, and reasonable computation and communication costs. Moreover, the security analysis demonstrates that the proposed protocol satisfies the security requirements in practical applications, and the proposed protocol is provably secure in the general security model.

1. Introduction

Rapid advances in wireless communication technologies bring convenience to our lives. With an increasing number of users and services, the single-server architecture authentication protocols can no longer meet people's various requirements [1]. Multiserver architecture authentication (MSAA) protocols have emerged and been widely used in the Internet of things, wireless sensor networks, smart grid, cloud computing, and mobile payment. Because MSAA protocols have better properties than single-server architecture authentication protocols [2, 3], it becomes a hot spot in current research.

However, due to the openness of the multiserver environment, an adversary can easily control communication channels and carries out many types of attacks such as intercept, modify, replay, and delay messages between multiple parties. For defending these attacks, researchers proposed many authentication protocols for the multiserver architecture that are using cryptographic methods to secure communication between different parties. New protocols also have lower computation and communication costs than the previous protocols. Currently, MSAA protocols can be divided into two types by whether it involves the registration center (RC) at the authentication phase. The first type is MSAA protocols with the RC involving at the authentication phase (MSAA1) [1, 2, 4–11], and the second type is MSAA protocols without RC involving at the authentication phase (MSAA2) [12–23]. In MSAA1 protocols,RC verifies every mutual authentication process, which makes it a bottleneck in MSAA1 protocols. The communication cost in MSAA1 protocols is significantly increased compared to MSAA2 protocols. To address these drawbacks, researchers proposed many MSAA2 protocols which have more efficiency and security than the existing MSAA1 protocols [14].

Currently, hierarchical authentication functionality is missing in the existing MSAA2 protocols. When a user registered at RC, he/she can authenticate with all registered service providers [24] and access to their services. However, there are many different users and service providers in this system, and the user's level is different from each other, and low-level users should not successfully authenticate with high-level service providers and access to their services. Besides, there should be some high-level service providers only providing service for some particular users such as VIP users. In general, the MSAA protocols with the hierarchical authentication functionality will have flexibility in managing

user authentication rights and access capabilities. The hierarchical authentication functionality has been achieved in the MSAA1 protocol [2, 4]. In the MSAA1 protocol, an RC is required at the authentication phase. Therefore, RC can verify the user's authentication rights to determine whether he/she can access to service providers that are at a particular level. However, MSAA1 protocols have several drawbacks such as unreasonable communication cost that we showed earlier, making the whole system inefficient. Suppose we apply the existing MSAA2 protocols to the above environment; there should be multiple RCs to manage users and service providers at different levels. Users and service providers need to store various certificates from different RCs. The missing of hierarchical authentication functionality in the existing MSAA2 protocols motivates us to design a new lightweight hierarchical authentication protocol for multiserver architecture.

The proposed protocol uses a self-constructed Merkle tree to achieve hierarchical authentication functionality. In the proposed protocol, a session key is established between service providers and users without involving RC; this significantly reduces communication cost and makes the authentication process faster. The proposed protocol can meet the security requirements of the multiserver architecture and is provably secure in general security model.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes preliminaries. In Section 4, we show the details of the proposed protocol. Section 5 gives out the formal security proof of the proposed protocol. Section 6 presents a comparison of the proposed protocol with other related protocols on security, computation, and communication costs. Section 7 concludes the paper.

2. Related Work

Li et al. [1] proposed a new multiserver architecture authentication (MSAA) protocol for cloud computing based on the identity-based model. Shao and Chin [4] proposed an authentication protocol for multiserver architecture but failed to resist the server spoofing and the impersonation attacks. He and Wang [5] constructed the first genuinely three-factor authentication protocol for the multiserver architecture, but their protocol is vulnerable to the known session-specific temporary information attack and impersonation attack. Odelu et al. [6] proposed an improved protocol to solve the security drawbacks in [5]. Xie et al. [7] proposed a two-factor authentication protocol. However, Xie's protocol cannot resist the lost smart card attack and the offline dictionary guessing attack. To address the drawbacks, Chandrakar and Om [8] proposed a new security-enhanced three-factor protocol. Feng et al. [9] proposed an enhanced biometrics-based authentication protocol that can provide user anonymity. Amin et al. [10] proposed a lightweight authentication protocol that has lower computational and communication costs. Cui et al. [11] proposed an efficient protocol that only uses nonce, exclusive-OR operation, and one-way hash function; their protocol greatly reduces the computation cost. However, in this kind of protocol, they all need the help of an online registration center to achieve mutual authentication, which increases the communication cost.

In order to solve the drawbacks in the first type protocol, Choi et al. [12] proposed the first MSAA protocol without the online registration center. Tseng et al. [13] proposed a list-free ID-based authentication protocol using bilinear pairings for the multiserver architecture. However, Tseng et al. [13] cannot provide credentials privacy and untraceability for users. Recently, Odelu et al. [14] and He et al. [15] proposed new protocols that reduce the computation and communication costs. Irshad et al. [16] found protocol in [17] cannot achieve desired security goals. Therefore, they proposed an improved multiserver authentication protocol for distributed mobile cloud computing services. Afterward, Xiong et al. [18] found protocol in [16] has unreasonable computation cost, so they proposed an enhanced protocol for distributed mobile cloud. At the same time, Xiong et al. [19] proposed a new lightweight anonymous authentication protocol to reduce computation and communication costs. Barman et al. [25] used fuzzy commitment approach to secure the information stored on personal device. Kumari et al. [20] proposed a concept of the fuzzy extractor to provide the proper matching of biometric patterns. Xu et al. [21] proposed a new protocol that provides untraceability. Jiang et al. [22] performed a security analysis to the protocol in [17], pointing out that it is vulnerable to the impersonation attack. Chatterjee et al. [23] proposed a biometric-based protocol using the chaotic map and enhanced the security for multiserver architecture. We summarize techniques, advantages, and disadvantages that the existing protocols used in Table 1.

3. Preliminaries

In this section, we introduce preliminaries of the proposed protocol.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be an additive cyclic group and a multiplicative cyclic group, both of them have a large prime order q. Let $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ denote a bilinear map. Suppose P is a generator of $\mathbb{G}_1$, g is a generator of $\mathbb{G}_2$. A bilinear map $\hat{e}$ has the following properties:

(i) Bilinearity: for all $P, Q \in \mathbb{G}_1$ and for all $a, b \in Z_q^*$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$

(ii) Computability: there exists an algorithm that can successfully compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$

(iii) Nondegeneracy: there exists $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$, where 1 is the identity element of $\mathbb{G}_2$

We list the hard problems that we used in the proposed protocol as follows:

(i) Discrete logarithm (DL) problem: given an element $x \in \mathbb{G}_2$, it is hard to compute $a \in Z_q^*$ such that $x = g^a$

(ii) Computational Diffie–Hellman (CDH) problem: given two elements $g^a, g^b \in \mathbb{G}_2$, it is hard to compute $g^{a \cdot b} \in \mathbb{G}_2$, where a and b are unknown and randomly chosen from Z_q^*

Table 1: Related work summaries.

Protocol	Technique	Advantage	Disadvantage
[1]	Identity-based	Lightweight and efficient	Cannot provide user anonymity
[4]	Identity-based	Provides user anonymity, resists server spoofing attack and impersonation attack, etc.	Cannot resist server spoofing attack and impersonation attacks
[5]	Biometrics-based	First truly three-factor authenticated scheme	Cannot resist known session-specific temporary attack and the impersonation attack
[6]	Biometrics-based	Provides secure authentication and resists passive and active attacks	Needs registration center online for authentication
[7]	Identity-based	Security enhanced and supports smart card revocation and password update without centralized storage	Cannot resist the lost smart card attack and the offline dictionary guessing attack
[8]	Biometrics-based	Efficient in terms of computation cost, communication cost, and resists smart card storage cost	High maintenance cost
[9]	Biometrics-based	Incurs low overhead, suitable for deployment at mobile devices	Needs registration center online for authentication
[10]	Two-factor-based	Security enhanced, lightweight, and efficient	Needs registration center online for authentication
[11]	Identity-based	Resists the server spoofing attack	Needs registration center online for authentication
[12]	Identity-based	Does not need registration center online for authentication	Cannot provide hierarchical authentication
[13]	Identity-based	Provides black/white list-free and simple revocation mechanism	Cannot provide credentials privacy and untraceability
[14]	Identity-based	Provides SK-security and strong credentials' privacy	Cannot provide hierarchical authentication
[15]	Identity-based	Uses the self-certified public key cryptography and has lower computation and communication costs	Cannot provide hierarchical authentication
[16]	Two-factor-based	Resists server spoofing attack, desynchronization attack, and denial-of-service attack	Cannot provide hierarchical authentication
[17]	Two-factor-based	Reduces authentication processing time required by communication and computation between cloud service providers and traditional trusted third-party service	Cannot resist service provider impersonation attack and has no user revocation facility
[18]	Biometrics-based	Provides three-factor security, user revocation, and reregistration	Cannot provide hierarchical authentication
[19]	Biometrics-based	User anonymity, perfect forward secrecy, and resistance to desynchronization attack	Cannot provide hierarchical authentication
[21]	Two-factor-based	Provides user untraceability and perfect forward security	Cannot provide hierarchical authentication
[23]	Biometric-based	Uses chaotic map to improve efficiency	Cannot provide hierarchical authentication

(iii) Modified Bilinear Inverse Diffie–Hellman with k value (k-mBIDH) problem [12]: given k elements $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ ($\alpha_i \in Z_q^*$) and $k+2$ elements $\{\tau \cdot P, \eta \cdot P, (1/(\tau+\alpha_1)) \cdot P, (1/(\tau+\alpha_2)) \cdot P, \ldots, (1/(\tau+\alpha_k)) \cdot P\}$ each of them is in $\mathbb{G}_1$, it is hard to compute $\widehat{e}(P,P)^{\eta/(\tau+\alpha)}$, where $\alpha \notin \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ and τ and η are two unknown elements in Z_q^*

A security system parameter generator used in the proposed protocol is introduced below.

Gen$(\cdot)$: the system parameter generator takes a security parameter n and outputs system parameters, a bilinear map, an elliptic curve, a multiplicative group, etc. Intuitively, the system parameters will be publicly known.

The notations used in the proposed protocol are listed in Table 2.

4. The Proposed Protocol

4.1. RC Initialization Phase. Registration center runs the generation function Gen(1^n) which takes a security parameter $n \in Z^+$ and outputs parameters as follows:

(1) RC chooses two bilinear map groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with a prime order q, the generator $P \in \mathbb{G}_1$ and $g = \widehat{e}(P,P) \in \mathbb{G}_2$, where $\widehat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ is a bilinear map.

(2) RC chooses cryptographic hash functions H_1: $\{0,1\}^* \longrightarrow Z_q^*$, H_2: $\mathbb{G}_2 \longrightarrow Z_q^*$, H_3: $\{0,1\}^* \longrightarrow \mathbb{G}_1$, H_4: $\{0,1\}^* \longrightarrow \{0,1\}^n$.

(3) RC chooses a random number $s \xleftarrow{R} Z_q^*$ as the master key, computes the corresponding public key $P_{\text{pub}} = sP \in \mathbb{G}_1$, and constructs an authentication right tree $\mathcal{T}$ as Figure 1. The detail of the tree will be described in Section 4.5. Finally, RC publishes $\{\mathbb{G}_1, \mathbb{G}_2, q, \widehat{e}, P, P_{\text{pub}}, g, H_1, H_2, H_3, H_4\}$.

4.2. User Registration Phase. If a user U_i wants to register with the registration center RC, the following steps will be executed. The main steps are provided in Table 3.

(1) U_i sends his/her identity ID_{U_i} to RC via a secure channel.

Table 2: Notations used in the proposed protocol.

Notation	Description
RC	The registration center
U_i	The i_{th} user
S_j	The j_{th} service provider
$\mathcal{A}$	The adversary
ID_{U_i}	The identity of i_{th} user
ID_{S_j}	The identity of j_{th} service provider
pw	The password of a user
b_i	The i_{th} user's personal biometrics
σ_i	The biometric key
e	The private key expire parameter
$\hat{r}_1, \hat{r}_2$	Random numbers from Z_q^*
sk	The session key
$\|$	Concatenation operation
$\mathcal{T}$	The authentication right tree
$\widehat{\mathcal{T}}$	The authentication right tree on service provider side
$a_i, \hat{a}_i$	The authentication right parameters
KR_i	The $i_t h$ node of the authentication right tree
L_i	The subnode of KR_i
$f(\cdot)$	The fuzzy-extractor generation procedure
$f(\cdot)^{-1}$	The deterministic reproduction procedure
$H_i(\cdot)$	Secure one-way hash functions
IDRL	ID revocation list

(2) RC selects an authentication parameter $KR_i \in \mathcal{T}$ and computes the U_i's private key $d_{U_i} = (1/(s + H_1(ID_{U_i}\|e\|KR_i))) \cdot P$, where e is the expire date of the private key and $\mathcal{T}$ is an authentication right tree. RC chooses a parameter $a_i \in \mathcal{T}$ and sends $\{d_{U_i}, a_i\}$ to U_i via a secure channel.

(3) U_i computes $(\sigma_i, \theta_i) \longleftarrow f(b_i)$ using the fuzzy-extractor generation procedure $f(\cdot)$ [26], where σ_i is a biometric key, θ_i is a public reproduction parameter, and b_i is his/her personal biometrics. U_i computes $A = d_{U_i} \oplus H_3(\text{pw}\|\sigma_i)$ and uses the widely implemented fuzzy-verifier technique [27, 28] to compute $B = H_4((H_4(ID_{U_i})\|H_4(\text{pw}\|\sigma_i))\text{mod}\, n_0)$, where pw is his/her password and n_0 is the integer that defines in [27]. Finally, U_i stores $\{a_i, \theta_i, A, B, e, f(\cdot), f^{-1}(\cdot), t, H_1, H_2, H_3, H_4\}$ on its mobile device, where t is the threshold in fuzzy extractor, $f(\cdot)$ is the probabilistic generation procedure for outputting σ_i, and θ_i, $f^{-1}(\cdot)$ is the deterministic reproduction procedure that can recover σ_i and θ_i from a new personal biometrics input.

4.3. Service Provider Registration Phase. If a service provider S_j wants to register with the RC, the following steps will be executed. The main steps are provided in Table 4.

(1) S_j sends his/her identity ID_{S_j} to RC via a secure channel.

(2) RC computes the private key $d_{S_j} = (1/(s + H_1(\text{ID}_{S_j}))) \cdot P$ for S_j and sends $\{d_{S_j}, \widehat{\mathcal{T}}\}$ to him via a secure channel, where $\widehat{\mathcal{T}}$ is an authentication right tree for service provider. We will describe the detail of $\widehat{\mathcal{T}}$ in Section 4.5.

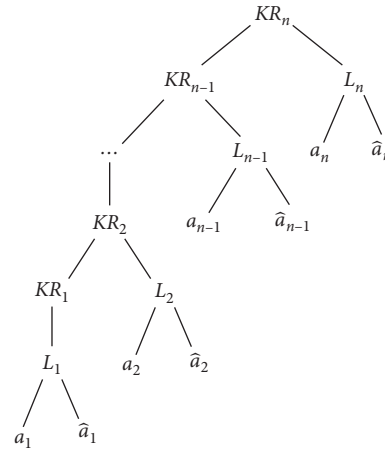

Figure 1: Authentication right tree.

(3) Finally, S_j saves $\{d_{S_j}, \widehat{\mathcal{T}}\}$.

4.4. User and Service Provider Authentication Phase. In this part, we show the mutual authentication phase between a user and a service provider without involving RC. The main steps are provided in Table 5.

(1) First, U_i inputs his/her biometrics b_i, identity ID_{U_i} and password pw into his/her mobile device. Mobile device computes $\sigma_i = f^{-1}(\theta_i, b_i)$ and $B^* = H_4((H_4(ID_{U_i})\|H_4(\text{pw}\|\sigma_i))\text{mod}\, n_0)$ and verifies the validity of inputted biometrics and password by computing $B^* \stackrel{?}{=} B$. If it holds, mobile device retrieves U_i's private key by computing $d_{U_i} = A \oplus H_3(\text{pw}\|\sigma_i)$ and temporarily saves ID_{U_i}. Then, U_i selects a temporary session secret $\overline{r}_1$, calculates $r_1 = H_1(\overline{r}_1\|d_{U_i})$, and computes $g_1 = g^{r_1}$, $C = r_1 \cdot (H_1(\text{ID}_{S_j})P + P_{\text{pub}})$ by using the identity of S_j. Next, U_i computes$D = H_1(\text{ID}_{U_i}\|e\|\text{ID}_{S_j}\|g_1)$, $E = (r_1 + D) \cdot d_{U_i}$, and $F = (a_i\|\text{ID}_{U_i}\|e\|\text{ID}_{S_j}) \oplus H_2(g^{r_1})$. Finally, U_i sends the login message $\{C, E, F\}$ to S_j.

(2) After receiving $\{C, E, F\}$, S_j checks whether ID_{U_i} is in IDRL; if ID_{U_i} is not in IDRL, S_j retrieves g_1 using his/her private key d_{S_j} as $g_1 = g^{r_1} = \hat{e}(C, d_{S_j})$. S_j retrieves $a_i, D, \text{ID}_{U_i}, e, \text{ID}_{S_j}$ by computing $F \oplus H_2(g_1)$. S_j computes $KR_i = H_4(KR_{i+1}\|L_i)$, where $L_i = H_4(a_i\|\hat{a}_i)$, and verifies $\hat{e}(E, H_1(\text{ID}_{U_i}\|e\|KR_i) \cdot P + P_{\text{pub}}) \stackrel{?}{=} g_1 \cdot g^D$. If both are equal, U_i is allowed to authenticate with S_j. Then, S_j selects a temporary session secret $\hat{r}_2$, then he/she calculates $r_2 = H_1(\hat{r}_2\|d_{S_j})$ and computes $g_2 = g^{r_2}$, and session key is set as $\text{sk} = H_2(g_1^{r_2}) = H_2(g^{r_1 r_2})$. Finally S_j calculates $G = H_4(sk\|g_1\|g_2\|\text{ID}_{S_j}\|C)$ and sends the message $\{g_2, G\}$ to U_i.

(3) Upon receiving $\{g_2, G\}$, U_i computes $\text{sk} = H_2(g_2^{r_1}) = H_2(g^{r_1 r_2})$and $G^* = H_4(\text{sk}\|g_1\|g_2\|\text{ID}_{S_j}\|C)$ and checks whether G and G^* are equal. If both are not equal, U_i aborts the session. Otherwise, U_i confirms S_j as a valid service provider and sets sk as session key between U_i and S_j.

Table 3: User registration phase.

Mobile user U_i	Registration center RC
$\xrightarrow[\text{secure channel}]{\text{ID}_{U_i}}$	
	Computes private key $d_{U_i} = (1/(s + H_1(ID_{U_i} \| e \| KR_i))) \cdot P$ and selects $a_i \in \mathcal{T}$
	$\xleftarrow[\text{secure channel}]{\{d_{U_i}, a_i\}}$
Computes $(\sigma_i, \theta_i) \longleftarrow f(b_i)A = d_{U_i} \oplus H_3(\text{pw} \| \sigma_i)$, $B = H_4((H_4(\text{ID}_{U_i}) \| H_4(pw \| \sigma_i)) \bmod n_0)$ and stores $\{a_i, \theta_i, A, B, e, f(\cdot), f^{-1}(\cdot), t, H_1, H_2, H_3, H_4\}$	

Table 4: Service provider registration phase.

Service provider S_j	Registration center RC
$\xrightarrow[\text{secure channel}]{\text{ID}_{S_j}}$	
	Computes $d_{S_j} = (1/(s + H_1(\text{ID}_{S_j}))) \cdot P$
	$\xleftarrow[\text{secure channel}]{\{d_{S_j}, \widehat{\mathcal{T}}\}}$
Stores $\{d_{S_j}, \widehat{\mathcal{T}}\}$	

Table 5: User and service provider authentication phase.

Mobile user U_i	Service provider S_j
$r_1 = H_1(\widehat{r_1} \| d_{U_i}), g_1 = g^{r_1}$ $C = r_1 \cdot (H_1(\text{ID}_{S_j}) \cdot P + P_{\text{pub}})$	
$D = H_1(\text{ID}_{U_i} \| e \| \text{ID}_{S_j} \| g_1)$	
$E = (r_1 + D) \cdot d_{U_i} F = (a_i \| \text{ID}_{U_i} \| e \| \text{ID}_{S_j}) \oplus H_2(g_1)$	
$\xrightarrow{\{C,E,F\}}$	
	Retrieves $g_1 = g^{r_1} = \widehat{e}(C, d_{S_j})(a_i \| \text{ID}_{U_i} \| e \| \text{ID}_{S_j}) = F \oplus H_2(g_1)$, computes $L_i = H_4(a_i \| \widehat{a}_i)$, $KR_i = H_4(KR_{i+1} \| L_i)\widehat{e}(E, H_1(\text{ID}_{U_i} \| e \| KR_i) \cdot P + P_{\text{pub}}) \stackrel{?}{=} g_1 \cdot g^D$, and accepts/rejects $r_2 = H_1(\widehat{r_2} \| d_{S_j}), g_2 = g^{r_2}$ sk $= H_2(g_1^{r_2})G = H_4(\text{sk} \| g_1 \| g_2 \| \text{ID}_{S_j} \| C)$
	$\xleftarrow{(G, g_2)}$
Computes sk $= H_2(g_2^{r_1})G^* = H_4(\text{sk} \| g_1 \| g_2 \| \text{ID}_{S_j} \| C)$, checks $G^* \stackrel{?}{=} G$, and accepts/rejects sk	

4.5. Tree Construction and Verification. The proposed protocol uses an authentication right tree $\mathcal{T}$ to store user's and service provider's hierarchical authentication information. $\mathcal{T}$ is a Merkle hash tree that was introduced by Merkle [29] in 1998. Merkle hash tree is a digital signature scheme that only uses a conventional encryption function to compute the digital signature, making it extremely efficient. In 2009, Satoshi proposed a peer-to-peer electronic cash system, as known as bitcoin [30]. Bitcoin system stores transactions in Merkle hash tree, which saves disk space, and this method can be used to verify transactions in each block. Therefore, we use a Merkle hash tree to construct our authentication right tree. In this part, we will show how we construct an authentication tree and how to verify a user's authentication right based on the rules of Merkle hash tree.

4.5.1. Tree Construction. First, we introduce the construction of the authentication right tree $\mathcal{T}$ as Figure 1. An authentication tree $\mathcal{T}$ contains the information of n different levels. The first level is the lowest level in the system, and the n_{th} level is the highest level in the system. Node KR_i denotes a user that has the authentication right which is from the first level to i_{th} level. Value stored in node KR_i is computed from the hash values of its left child node KR_{i-1} and right child node L_i as $KR_i = H_4(KR_{i-1} \| L_i)$. The calculation of node KR_1 as $KR_1 = H_4(L_1)$ is different from other KR nodes. If a user is at the first level, KR_1 is embed in his/her private key, and he/she can only access to first level service providers in this system. If user is at i_{th} level, KR_i is embed in his/her private key, and he/she can access to service providers which are from first to i_{th} levels. The right

child node L_i is an intermediate variable, which prepares for calculating KR_i. Value stored in node L_i is computed from the hash values of its left leaf node a_i and right leaf node $\widehat{a}_i$ as $L_i = H_4(a_i \| \widehat{a}_i)$. Leaf node a_i and $\widehat{a}_i$ are two 160-bit random strings that are stored on user and service provider separately. If a user is at i_{th} level, number i is stored in the last $\log_2^n$ bits, and the first $(160 - \log_2^n)$ bits should be a random string.

4.5.2. Tree Stored on Service Provider. The authentication right tree $\widehat{\mathcal{T}}$ stored on the service provider has a little different from $\mathcal{T}$. Service provider uses $\widehat{\mathcal{T}}$ to verify user's authentication right. The scale of $\widehat{\mathcal{T}}$ stored on each service provider is based on the level of service provider. As we mentioned above, n_{th} level is the highest level in the system. If service provider is at n_{th} level, he/she only provides service for n_{th} level user, so he/she only needs to save the authentication right tree $\widehat{\mathcal{T}}$ as Figure 2. If service provider is at the j_{th} level, he/she can provide service for user that is from j_{th} to n_{th} level. Therefore, he/she needs to save $\widehat{a}_j$ to $\widehat{a}_n$ and KR_{j-1} to KR_n as Figure 3, where the symbol "?" denotes the node that service provider does not have.

4.5.3. Authentication Right Verification. When an i_{th} level user wants to access a j_{th} level service provider, where $i \geq j$, user sends a_i to service provider, and when service provider received authentication parameter a_i, he/she checks the last $\log_2^n$ bits of a_i to get user's level and finds $\widehat{a}_i$. Service provider computes the value of L_i as $L_i = H_4(a_i \| \widehat{a}_i)$ and the value of KR_i as $KR_i = H_4(KR_{i-1} \| L_i)$. Then service provider verifies user's authentication right by calculating $\widehat{e}(E, H_1(\text{ID}_{U_i} \| e \| KR_i) \cdot P + P_{\text{pub}}) \stackrel{?}{=} g_1 \cdot g^D$. If the equation holds, service provider continues. Otherwise, he/she aborts the session. For instance, if a 10_{th} level user wants to access to a 5_{th} level service provider, user sends a_{10} to service provider, and when service provider received authentication parameter a_{10}, he/she checks the last $\log_2^n$ bits of a_{10} to get user's level and finds $\widehat{a_{10}}$. Service provider computes $L_{10} = H_4(a_{10} \| \widehat{a_{10}})$ and $KR_{10} = H_4(KR_9 \| L_{10})$. Service provider verifies user's authentication right by calculating $\widehat{e}(E, H_1(\text{ID}_{U_i} \| e \| KR_{10}) \cdot P + P_{\text{pub}}) \stackrel{?}{=} g_1 \cdot g^D$. If the equation holds, service provider continues. Otherwise, he/she aborts the authentication.

4.6. The User Revocation and Reregistration Phase. Revocation and reregistration has been used in many protocols [31, 32]. In this part, we describe user revocation and reregistration. When a user U_i lost his/her smart card, he/she needs to reregister. U_i submits his/her personal information to RC, and then, RC verifies U_i's personal information and checks the expire date of the private key d_{U_i}. If d_{U_i} has already expired, RC issues U_i a new private key with a new expire date. If d_{U_i} has not expired, RC issues U_i a new ID and a new private key with the same expire date as the lost smart card. RC adds the lost ID_{U_i} to its ID revocation list (IDRL) and board casts ID_{U_i} to all service providers. After received ID_{U_i}, service providers save it into their local storage.

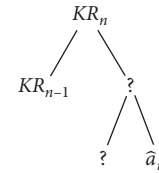

Figure 2: Authentication right tree on n_{th} level service provider.

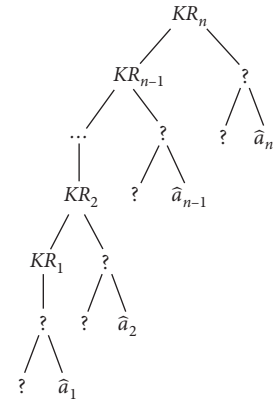

Figure 3: Authentication right tree on j_{th} level service provider.

5. Security Proof

In this section, we analyze the security of our protocol. First, we present a security model for our protocol, which is based on Bellare–Rogaway (BR) model [33] and CK-adversary model [34], and we use Zipf's law [35] to enhance the security of the base model. Second, we show that the security of the proposed protocol is based on the hardness of mathematical problems. Third, we show that our protocol satisfies security requirements.

5.1. Security Model. We propose a security model for the proposed protocol based on literature studies [5, 15, 27, 33, 36]. There are U_i and S_j at the authentication phase of the proposed protocol. The security of the proposed protocol is defined by a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Let Π_Λ^l denote the l_{th} instance of the participant of $\Lambda \in \{U_i, S_j\}$, respectively. In this game, we describe the capabilities of $\mathcal{A}$ that is defined in the literature [27] as follows:

(i) $\mathcal{A}$ can enumerate offline all the items in the Cartesian product $D_{\text{id}} \star D_{\text{pw}}$ within polynomial time, where D_{pw} and D_{id} denote the password space and the identity space, respectively

(ii) $\mathcal{A}$ has the capability of somehow learning the victim's identity when evaluating security strength (but not privacy provisions) of the protocol

(iii) $\mathcal{A}$ is in full control of the communication channel between the protocol participants

(iv) $\mathcal{A}$ may either (i) learn the password of a legitimate user via malicious card reader or (ii) extract the sensitive parameters in the card memory by side-channel attacks, but cannot achieve both

(v) $\mathscr{A}$ can learn previous session keys

(vi) $\mathscr{A}$ has the capability of learning server's longtime private keys only when evaluating the resistance to eventual failure of the server (e.g., forward secrecy)

$\mathscr{A}$ can issue queries to $\mathscr{C}$ and get answers from it as follows:

(i) $H_i(q_j)$: at any time, $\mathscr{A}$ issues query q_j where q_j can be any string, and $\mathscr{C}$ picks a random number $r_j \in Z_q^*$ and stores $\langle q_j, r_j \rangle$ into list $\mathscr{H}_i^{\text{list}}$, where $i \in \{1, 2, 3, 4\}$ and $j \in \{\text{poly}(n)\}$. Finally, $\mathscr{C}$ sends r_j to $\mathscr{A}$.

(ii) ExtractUID (ID_{U_i}): $\mathscr{A}$ issues queries of user identity ID_{U_i}, and $\mathscr{C}$ generates users private key d_{U_i} and stores $\langle \text{ID}_{U_i}, d_{U_i} \rangle$ into list $\mathscr{E}_{d_{U_i}}^{\text{list}}$.

(iii) ExtractSID (ID_{S_j}): $\mathscr{A}$ issues queries of service provider's identity ID_{S_j}, and $\mathscr{C}$ generates service provider's private key d_{S_j} and stores $\langle \text{ID}_{S_j}, d_{S_j} \rangle$ into list $\mathscr{E}_{d_{S_j}}^{\text{list}}$.

(iv) Send (Π_Λ^l): $\mathscr{A}$ issues query of the message m, and $\mathscr{C}$ runs the protocol and returns the result to $\mathscr{A}$.

(v) SKReveal (Π_Λ^l): $\mathscr{A}$ issues query, and $\mathscr{C}$ returns the session key produced in Π_Λ^l to $\mathscr{A}$.

(vi) CorruptUID (ID_{U_i}): $\mathscr{A}$ issues the query of user's identity ID_{U_i}, and $\mathscr{C}$ returns user's private key d_{U_i} to $\mathscr{A}$.

(vii) CorruptSID (ID_{S_j}): $\mathscr{A}$ issues the query of service provider's identity ID_{S_j}, and $\mathscr{C}$ returns service provider's private key d_{S_j} to $\mathscr{A}$.

(viii) Test (Π_Λ^l): when $\mathscr{A}$ issues the query, $\mathscr{C}$ flips a random coin $b \in \{0, 1\}$. If $b = 1$, $\mathscr{C}$ sends session key in Π_Λ^l to $\mathscr{A}$; otherwise, $(b = 0)$, and $\mathscr{C}$ picks a random number with the same length of session key and sends it to $\mathscr{A}$.

After issuing the queries above, $\mathscr{A}$ outputs b', where b' is about the coin b produced in Test (Π_Λ^l)-query. $\mathscr{A}$ violates the authentication key agreement (AKA) of the proposed protocol Σ, if $\mathscr{A}$ can guess b correctly. We define $\mathscr{A}$'s advantage in attacking the proposed protocol Σ as $\text{Adv}_\Sigma^{\text{AKA}}(\mathscr{A}) = |2\Pr[b = b'] - 1|$.

Definition 1 (AKA-Secure). The proposed protocol Σ is authentication key agreement secure (AKA-Secure) if $\text{Adv}_\Sigma^{\text{AKA}}(\mathscr{A}) = |2\Pr[b = b'] - 1|$ is negligible for any polynomial-time adversary $\mathscr{A}$.

$\mathscr{A}$ violates the mutual authentication of the proposed protocol Σ, if $\mathscr{A}$ can generate a legal login message or a legal response message. Let $E_{\{U,S\}}$ and $E_{\{S,U\}}$ denote the events that $\mathscr{A}$ generates a legal login message and a legal response message. We define the advantage of $\mathscr{A}$ attacking the mutual authentication of the proposed protocol Σ as $\text{Adv}_\Sigma^{\text{MA}}(\mathscr{A}) = \Pr[E_{\{U,S\}}] + \Pr[E_{\{S,U\}}]$.

Definition 2 (MA-Secure). The proposed protocol Σ is mutual authentication secure (MA-Secure) if $\text{Adv}_\Sigma^{\text{MA}}(\mathscr{A})$ is negligible for any polynomial-time adversary $\mathscr{A}$.

5.2. Proof of Security. In this part, we show the proposed protocol Σ for multiserver architecture is AKA-secure and MA-secure in the security model we described above.

Lemma 1. *No polynomial-time adversary $\mathscr{A}$ can forge a legal login message with a nonnegligible probability ϵ.*

Proof. Suppose the adversary $\mathscr{A}$ forges a legal login message with a nonnegligible probability ϵ. We show there is a challenger $\mathscr{C}$ who can solve the discrete logarithm (DL) problem with a nonnegligible probability.

Given an instance (g, g^s) of the DL problem, the aim of challenger $\mathscr{C}$ is to compute $s \in Z_q^*$, and $\mathscr{C}$ sends the system parameters $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{\text{pub}}, g, H_1, H_2, H_3, H_4\}$ to $\mathscr{A}$. $\mathscr{C}$ randomly selects ID_{U_i} and answers $\mathscr{A}$'s queries according to the following description:

(i) $H_i(q_j)$: $\mathscr{C}$ maintains a list H_i^{list} initialized empty. Upon receiving the query q_j, $\mathscr{C}$ checks if $\langle q_j, r_j \rangle$ exists in H_i^{list}. If yes, $\mathscr{C}$ sends r_j to $\mathscr{A}$; otherwise, $\mathscr{C}$randomly picks $r_j \in Z_q^*$ and stores $\langle q_j, r_j \rangle$ in H_i^{list} then sends r_j to $\mathscr{A}$, where $j \in \{\text{poly}(n)\}$.

(ii) ExtractUID (ID_{U_i}): $\mathscr{C}$ maintains a list $\mathscr{E}_{d_{U_i}}^{\text{list}}$ initialized empty. When receiving the query ID_{U_i}, $\mathscr{C}$checks if $\langle \text{ID}_{U_i}, d_{U_i} \rangle$ exists in $\mathscr{E}_{d_{U_i}}^{\text{list}}$ then $\mathscr{C}$ send d_{U_i} to $\mathscr{A}$; otherwise, $\mathscr{C}$ executes the operations as follows:

(1) If $\text{ID}_{U_i} = \text{ID}_{U_{ch}}$, $\mathscr{C}$ sets $H_1(\text{ID}_{U_i}) \longleftarrow \alpha$, $d_{U_i} \longleftarrow \perp$ and stores $\langle \text{ID}_{U_i}, \alpha \rangle$ into H_1^{list}, $\langle \text{ID}_{U_i}, d_{U_i} \rangle$ into $\mathscr{E}_U^{\text{list}}$

(2) Otherwise if $\text{ID}_{U_i} \neq \text{ID}_{U_{ch}}$, $\mathscr{C}$ randomly selects $\alpha_i \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, sets $H_1\{\text{ID}_{U_{ch}}\} \longleftarrow \alpha_i, d_{U_i} \longleftarrow (1/(s + \alpha_i)) \cdot P$, and stores $\langle \text{ID}_{U_i}, \alpha_i \rangle$ in H_1^{list}, $\langle \text{ID}_{U_i}, d_{U_i} \rangle$ in $\mathscr{E}_{d_{U_i}}^{\text{list}}$

(iii) ExtractSID (ID_{S_j}): $\mathscr{C}$ maintains a list $\mathscr{E}_{d_{S_j}}^{\text{list}}$ initialized empty. When receiving the query ID_{S_j}, $\mathscr{C}$ checks if $\langle \text{ID}_{S_j}, d_{S_j} \rangle$ exists in $\mathscr{E}_{d_{S_j}}^{\text{list}}$. If yes, $\mathscr{C}$ send d_{S_j} to $\mathscr{A}$; otherwise, $\mathscr{C}$ randomly picks $r_{d_{S_j}} \in Z_q^*$ and computes $d_{S_j} = (1/(s + r_{d_{S_j}})) \cdot P$. $\mathscr{C}$ stores $\langle \text{ID}_{S_j}, d_{S_j} \rangle$ into $\mathscr{E}_{d_{S_j}}^{\text{list}}$, $\langle \text{ID}_{S_j}, r_{d_{S_j}} \rangle$ into H_1^{list} and sends d_{S_j} to $\mathscr{A}$.

(iv) Send (Π_Λ^l): $\mathscr{C}$ checks if Λ and U_{ch} are equal; if yes, $\mathscr{C}$ aborts the game; otherwise, $\mathscr{C}$ operates according to protocol Σ.

(v) SKReveal (Π_Λ^l): after received the query, $\mathscr{C}$ sends session key produced in Π_Λ^l to $\mathscr{A}$.

(vi) CorruptUID (ID_{U_i}): $\mathscr{C}$ searches for $\langle \text{ID}_{U_i}, d_{U_i} \rangle$ in $\mathscr{E}_{d_{U_i}}^{\text{list}}$ and returns d_{U_i} to $\mathscr{A}$.

(vii) CorruptSID (ID_{S_j}): $\mathscr{C}$ searches for $\langle ID_{S_j}, d_{S_j} \rangle$ in $\mathscr{E}_{d_{S_j}}^{list}$ and returns d_{S_j} to $\mathscr{A}$.

(viii) Test (Π_Λ^l): $\mathscr{C}$ randomly picks a number with the same length of session key and sends it to $\mathscr{A}$.

At last, $\mathscr{A}$ outputs a legal login message $\{C, E, F\}$ corresponding to user's identity ID_{U_i}. If $\text{ID}_{U_i} \neq \text{ID}_{U_{ch}}$, $\mathscr{C}$ aborts the game. Based on the forking lemma [37], $\mathscr{A}$ can output

another legal login message $(C, E\prime, F\prime)$. Because the login messages is legal, we get the following two equations, g_{U_i} is computed by rising g to the power of a random number chose by $\mathscr{A}$:

$$\begin{aligned}\widehat{e}\left(E, H_1\left(\mathrm{ID}_{U_i}\|e\|KR_i\right)\cdot P + P_{\mathrm{pub}}\right) &= g_{U_i}\cdot g^{D},\\ \widehat{e}\left(E', H_1\left(\mathrm{ID}_{U_i}\|e\|KR_i\right)\cdot P + P_{\mathrm{pub}}\right) &= g_{U_i}\cdot g^{D'}.\end{aligned} \tag{1}$$

Based on the two equations above, we get the following equations:

$$\frac{\widehat{e}\left(E, H_1\left(\mathrm{ID}_{U_i}\|e\|KR_i\right)\cdot P + P_{\mathrm{pub}}\right)}{\widehat{e}\left(E', H_1\left(\mathrm{ID}_{U_i}\|e\|KR_i\right)\cdot P + P_{\mathrm{pub}}\right)} = \frac{g_{U_i}\cdot g^{D}}{g_{U_i}\cdot g^{D'}} = g^{D-D'}, \tag{2}$$

$\mathscr{C}$ outputs $(D - D')^{-1}\cdot(E - E')$ as the solution to the given DL problem. The probability that $\mathscr{C}$ can solve the DL problem is described as follows:

(i) E_1: $\mathscr{C}$ dose not abort in the any Send-queries

(ii) E_2: $\mathscr{A}$ outputs a legal login request

(iii) E_3: ID_{U_i} and ID_{U_I} are equal

Let l denote the number of bits in biometric data, q_{Send} and q_{H_1} denote the number of Send-queries and H_1-queries executed in the game, C' and s' are the Zipf's parameters [35], and l is the length of biometric information. We can get $\Pr[E_1] \geq (1 - (1/(q_{\mathrm{Send}} + 1)))^{q_{\mathrm{Send}}}$, $\Pr[E_2 \mid E_1] \geq \varepsilon$, and $\Pr[E_3 \mid E_1 \wedge E_2] \geq (1/q_{H_1})$.

Therefore, the nonnegligible probability that $\mathscr{C}$ can solve the DL problem is given by

$$\begin{aligned}\Pr\left[E_1 \wedge E_2 \wedge E_3\right] &= \Pr\left[E_3 \mid E_1 \wedge E_2\right],\\ \Pr\left[E_2 \mid E_1\right]\cdot\Pr\left[E_1\right] &\geq \frac{\left(1 - \left(1/\left(q_{\mathrm{Send}} + 1\right)\right)\right)^{q_{\mathrm{Send}}}}{q_{H_1}}\cdot\varepsilon\\ &\quad + \max\left\{C'\cdot q_{\mathrm{Send}}^{s'}, \frac{q_{\mathrm{Send}}}{2^{l}}\right\}.\end{aligned} \tag{3}$$

This contradicts with the hardness of the DL problem. Therefore, we get that no polynomial-time adversary against the proposed MSAA protocol can forge a legal login message with a nonnegligible probability. □

Lemma 2. *No polynomial-time adversary $\mathscr{A}$ can forge a legal response message with a nonnegligible probability.*

Proof. Suppose the adversary $\mathscr{A}$ forges a legal response message with a nonnegligible probability ε. We show there is a challenger $\mathscr{C}$ who can solve the k-mBIDH problem with a nonnegligible probability.

Given an instance $(P, y\cdot P, z\cdot P, (1/(y+\alpha_1))\cdot P, (1/(y+\alpha_2))\cdot P, \ldots, (1/(y+\alpha_k))\cdot P \in \mathbb{G}_1)$ of the k-mBIDH problem, the aim of challenger $\mathscr{C}$ is to compute $\widehat{e}(P,P)^{s/(y+\alpha)}$; he picks a random number $x \in Z_q^*$ and computes $x\cdot P$, $y\cdot P$, and sends the system parameters $\{\mathbb{G}_1, \mathbb{G}_2, q, e, P, P_{\mathrm{pub}}, g, H_1, H_2, H_3, H_4\}$ to $\mathscr{A}$. $\mathscr{C}$ randomly selects ID_{U_i} and answers $\mathscr{A}$'s queries according to the following description:

(i) $H_i(q_j)$: $\mathscr{C}$ maintains a list H_i^{list} initialized empty. Upon receiving the query q_j, $\mathscr{C}$ checks if $\langle q_j, r_j\rangle$ exists in H_i^{list}. If yes, $\mathscr{C}$ sends r_j to $\mathscr{A}$; otherwise, $\mathscr{C}$ randomly picks $r_j \in Z_q^*$ and stores $\langle q_j, r_j\rangle$ in H_i^{list} then sends r_j to $\mathscr{A}$, where $j \in \{\mathrm{poly}(n)\}$.

(ii) ExtractUID (ID_{U_i}): $\mathscr{C}$ maintains a list $\mathscr{E}_{d_{U_i}}^{\mathrm{list}}$ initialized empty. When receiving the query ID_{U_i}, $\mathscr{C}$ checks if $\langle \mathrm{ID}_{U_i}, d_{U_i}\rangle$ exists in $\mathscr{E}_{d_{U_i}}^{\mathrm{list}}$ and then $\mathscr{C}$ sends d_{U_i} to $\mathscr{A}$; otherwise, $\mathscr{C}$ randomly picks $r_{d_{U_i}} \in Z_q^*$ and computes $d_{U_i} = (1/(s + r_{d_{U_i}}))\cdot P$. $\mathscr{C}$ stores $\langle \mathrm{ID}_{U_i}, d_{U_i}\rangle$ in $\mathscr{E}_{d_{U_i}}^{\mathrm{list}}$, $\langle \mathrm{ID}_{U_i}, r_{d_{U_i}}\rangle$ in H_1^{list} and sends d_{U_i} to $\mathscr{A}$.

(iii) ExtractSID (ID_{S_j}): $\mathscr{C}$ maintains a list $\mathscr{E}_{d_{S_j}}^{\mathrm{list}}$ initialized empty. When receiving the query ID_{S_j}, $\mathscr{C}$ checks if $\langle \mathrm{ID}_{S_j}, d_{S_j}\rangle$ exists in $\mathscr{E}_{d_{S_j}}^{\mathrm{list}}$ and then $\mathscr{C}$ sends d_{S_j} to $\mathscr{A}$; otherwise, $\mathscr{C}$ executes the operations as follows:

(1) If $\mathrm{ID}_{S_j} = \mathrm{ID}_{S_{ch}}$, $\mathscr{C}$ sets $H_1\{\mathrm{ID}_{U_i}\} \longleftarrow \alpha$ and stores $\langle \mathrm{ID}_{S_j}, \alpha\rangle$ into H_1^{list}, $\langle \mathrm{ID}_{S_j}, \perp\rangle$ into $\mathscr{E}_{d_{S_j}}^{\mathrm{list}}$

(2) Otherwise if $\mathrm{ID}_{S_j} \neq \mathrm{ID}_{S_{ch}}$, $\mathscr{C}$ randomly selects $\alpha_i \in \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, sets $H_1\{\mathrm{ID}_{S_j}\} \longleftarrow \alpha$, and stores $\langle \mathrm{ID}_{S_j}, \alpha_i\rangle$ in H_1^{list}, $\langle \mathrm{ID}_{S_j}, \alpha_i, (1/(s+\alpha_i))\cdot P\rangle$ in $\mathscr{E}_{d_{S_j}}^{\mathrm{list}}$

(iv) Send (Π_Λ^l): $\mathscr{C}$ checks if Λ and S_{ch} are equal; if yes, $\mathscr{C}$ aborts the game; otherwise, $\mathscr{C}$ operates according to the proposed protocol Σ.

Finally, $\mathscr{A}$ outputs a response message corresponding to identity ID_{S_j}. $\mathscr{C}$ outputs g_1 as the solution of k-mBIDH problem. The probability that $\mathscr{C}$ can solve the k-mBIDH problem is described as follows:

(i) E_1: $\mathscr{C}$ dose not abort in any Send-queries

(ii) E_2: C outputs a legal response message

(iii) E_3: ID_{S_j} and $\mathrm{ID}_{S_{ch}}$ are equal

Let $q_{\mathrm{Send}}, q_{H_1}$, and q_{H_2} denote the number of Response-query, H_1-query, and H_2-query in the game. We can get $\Pr[E_1] \geq (1 - (1/(q_{\mathrm{Send}} + 1)))^{q_{\mathrm{Send}}}$, $\Pr[E_2 \mid E_1] \geq \varepsilon$, and $\Pr[E_3 \mid E_1 \wedge E_2] \geq (1/(q_{H_1}\cdot q_{H_2}))$. Therefore, the nonnegligible probability that $\mathscr{C}$ can solve the k-mBIDH problem is given by

$$\begin{aligned}\Pr\left[E_1 \wedge E_2 \wedge E_3\right] &= \Pr\left[E_3 \mid E_1 \wedge E_2\right]\cdot\Pr\left[E_2 \mid E_1\right]\cdot\Pr\left[E_1\right]\\ &\geq \frac{\left(1 - \left(1/\left(q_{\mathrm{Send}} + 1\right)\right)\right)^{q_{\mathrm{Send}}}}{q_{H_1}\cdot q_{H_2}}\cdot\varepsilon.\end{aligned} \tag{4}$$

This contradicts with the hardness of the k-mBIDH problem. Therefore, we get that no polynomial-time adversary against the proposed MSAA protocol can forge a legal response message with a nonnegligible probability. □

Theorem 1. *The proposed protocol is MA-secure if the DL problem and the k-mBIDH problem are hard.*

Proof. Based on Lemmas 1 and 2, we get no polynomial-time adversary can forge a legal login message or a legal response message if the DL problem and the k-mBIDH problem are hard. Therefore, we get the proposed protocol is MA-secure. □

Theorem 2. *The proposed protocol is AKA-secure if the CDH problem is hard.*

Proof. Suppose $\mathscr{A}$ guesses b correctly in Test-query with a nonnegligible probability ϵ, then $\mathscr{C}$ can solve the CDH problem with a nonnegligible probability.

Let E_{sk} denote the event that $\mathscr{A}$ gets the correct session key. Since the probability that $\mathscr{A}$ correctly guesses the value b is at least 1/2, we can get $\Pr[E_{\text{sk}}] \geq (\varepsilon/2)$.

Let E_{TU} and E_{TS} denote the events that $\mathscr{A}$ uses in the Test-query to a user's instance and a service provider's instance, respectively. Let $E_{\{U,S\}}$ denote the event that $\mathscr{A}$ can violate the user to service provider authentication. We get the following two equations:

$$\begin{aligned}\frac{\varepsilon}{2} &\leq \Pr[E_{\text{sk}}] = \Pr[E_{\text{sk}} \wedge E_{TU}] + \Pr[E_{\text{sk}} \wedge E_{TS}] \\ &\wedge E_{\{U,S\}} + \Pr\left[E_{\text{sk}} \wedge E_{TS} \wedge \neg E_{\{U,S\}}\right] \leq \Pr[E_{\text{sk}} \wedge E_{TU}] + \Pr\left[E_{\{U,S\}}\right] + \Pr\left[E_{\text{sk}} \wedge E_{TS} \wedge \neg E_{\{U,S\}}\right], \\ \Pr[E_{sk} \wedge E_{TU}] &+ \Pr[E_{sk} \wedge E_{TS} \wedge \neg E\{U,S\}] \geq \frac{\varepsilon}{2} - \Pr\left[E_{\{U,S\}}\right].\end{aligned} \tag{5}$$

Since $\Pr[E_{TS} \wedge E_{\{U,S\}}] = \Pr[E_{TU}]$, we get

$$\Pr\left[E_{TS} \wedge E_{\{U,S\}}\right] \geq \frac{\varepsilon}{4} - \frac{\Pr\left[E_{\{U,S\}}\right]}{2}. \tag{6}$$

We get the probability as follows:

$$\Pr\left[\text{sk} = H_2\left(g^{r_1 r_2}\right) \mid r_1, r_2 \longleftarrow Z_q^*\right] \geq \frac{\varepsilon}{4} - \frac{\Pr\left[E_{\{U,S\}}\right]}{2}. \tag{7}$$

According to the proof of Lemma 1, we get $\Pr[E_{\{U,S\}}]$ is negligible. However, $(\varepsilon/4) - ((\Pr[E_{\{U,S\}}])/2)$ is nonnegligible, suppose that $x = g^{r_1}$, $y = g^{r_2}$ where $r_1, r_2 \in Z_q^*$. Given an instance (x, y) of the CDH problem, $\mathscr{A}$ computes $z = g^{r_1 \cdot r_2}$ with a nonnegligible probability $(\varepsilon/4) - ((\Pr[E_{\{U,S\}}])/2)$. Therefore, $\mathscr{C}$ can use $\mathscr{A}$ to solve the CDH problem with a nonnegligible probability. This contradicts with the hardness of the CDH problem. Therefore, we can conclude that the proposed protocol is AKA-secure if the CDH problem is hard. □

5.3. Security Requirements Analysis. We briefly show the proposed protocol satisfies the security requirements as follows:

(i) Single registration: according to the specification of the proposed protocol, a user registers at the registration center once, and he/she can log into registered service providers, which is at a specific level. Therefore, the proposed protocol can provide single registration.

(ii) Mutual authentication: two lemmas described above show that the adversary against the proposed protocol cannot produce a valid login or response message. Then, ID_{U_i} and ID_{S_j} can authenticate with the participant by checking the legality of the received response message and login message, respectively. Therefore, the proposed protocol can support mutual authentication.

(iii) User anonymity: according to the proposed protocol, the user's identity ID_{U_i} is only in the message $F = (D \| \text{ID}_{U_i} \| e \| \text{ID}_{S_j}) \oplus H_2(g^{r_1})$. To get ID_{U_i}, adversary needs to compute g^{r_1} from $C = r_1 \cdot (H_1(ID_{S_j}) \cdot P + P_{\text{pub}})$, and it turns out the adversary need to solve the k-mBIDH problem. Then, we know that the proposed protocol can provide user anonymity as long as k-mBIDH problem is hard.

(iv) Untraceability: according to the proposed protocol, user generates a new random number $r_1 \in Z_q^*$ to compute $C = r_1 \cdot (H_1(\text{ID}_{S_j}) \cdot P + P_{\text{pub}})$, g^{r_1}, $F = (a_i \| D \| \text{ID}_{U_i} \| e \| \text{ID}_{S_j}) \oplus H_2(g^{r_1})$. Due to the randomness of r_1, adversary cannot find any relation of messages sent by the user and cannot trace the user's behavior. Therefore, the proposed protocol can provide untraceability.

(v) Session key agreement: according to the proposed protocol, both two participants calculate session key $\text{sk} = H_2(g^{r_1 r_2})$, which can be used in future communications. Therefore, the proposed protocol can provide session key agreement.

(vi) Perfect forward secrecy: assume the adversary steals both private keys of the user and the service provider. We also assume that the adversary intercepts C, E, F, G, g_2 sent between the user and the service provider. Using the service provider's private key, the adversary can compute $g_1 = \hat{e}(C, d_{S_j}) = g^{r_1}$. To get session key $\text{sk} = H_2(g^{r_1 r_2})$, the adversary must to compute $g_1^{r_2} = g_2^{r_1} = g^{r_1 r_2}$ where $g_1 = g^{r_1}$, $g_2 = g^{r_2}$. Thus, adversary must solve the CDH problem. Then, the proposed protocol can provide the perfect forward secrecy, since the CDH problem is hard.

(vii) No smart card lose attack [20]: assume the adversary steals the user's device. By using the side-

channel attack, the adversary can extract the data $B = H_4(\mathrm{ID}_{U_i} \| H_4(\mathrm{pw} \| \sigma_i))$, $A = d_{U_i} \oplus H_3(\mathrm{pw} \| \sigma_i)$. The adversary can guess password pw, but he/she cannot verify its correctness because we have implemented the fuzzy-verifier technique [27, 28]. The adversary cannot get the user's password, so he cannot get the user's private key d_{U_i}. Therefore, adversaries cannot impersonate the user to the service provider, and the proposed protocol can resist smart card lose attack.

(viii) No password verifier table: according to the proposed protocol, both two participants need to store their private keys, and the registration center needs no password verifier table. Thus, the proposed protocol provides no password verifier table.

(ix) No online registration center: according to the proposed protocol, two participants can authenticate with each other without the help of the registration center. Thus, the proposed protocol does not need an online registration center.

(x) Hierarchical authentication: this security requirement only satisfied by the proposed protocol. The hierarchical information KR_i is embedded in the user's private key $d_{U_i} = (1/(s + H_1(\mathrm{ID}_{U_i} \| e \| KR_i))) \cdot P$, when authentication is with a service provider, service provider will check the user's authentication right by computing whether the equation $\hat{e}(E, H_1(\mathrm{ID}_{U_i} \| e \| KR_i) \cdot P + P_{\mathrm{pub}}) \stackrel{?}{=} g_1 \cdot g^D$ holds.

(xi) The resistance of various attacks: the proposed protocol can resist the insider attack, the replay attack, the man-in-the-middle attack, etc. We briefly describe it as follows:

(1) Temporary information attack: if the adversary gets the temporary information $\hat{r_1}, \hat{r_2}$, he/she has no ability to derive the user's secret key from $(r_1 + D) \cdot d_{U_i}$ because the exponential of g is composed of two values: one is session temporary secret $\hat{r_1}$ and other is the private key of the user d_{U_i}. Therefore, the proposed protocol is secure against temporary information attack.

(2) Insider attack: suppose an insider in the system gets the user's information $\mathrm{ID}_{U_i}, H_3(\mathrm{pw} \| \sigma_i)$. The adversary can guess a password pw. However, he/she cannot verify its correctness because user's password is protected by the secure hash function and the biometric key σ_i. Thus, the insider cannot get user's password, and the proposed protocol withstands the insider attack.

(3) User impersonation attack [38, 39]: according to the proof of Lemma 1, we conclude that no adversary can forge a legal login message without the user's private key. Thus, the service provider can find out about the attack by verifying the validity of the received login message. Therefore, the proposed protocol can resist the user impersonation attack.

(4) Server spoofing attack: according to the proof of Lemma 2, we know that no adversary can generate a legal response message without the service provider's private key. Therefore, users can find out about the attack by verifying the validity of the received response message. Therefore, the proposed protocol can resist the server spoofing attack.

(5) Modification attack: according to the proof of Lemma 1, we know that C, E, F is a digital signature of the login message and no polynomial-time adversary can forge a legal one. The service provider can find any modification by checking if the equation $g_1 = g^{r_1} = \hat{e}(C, d_{S_j})$ and $\hat{e}(E, H_1(\mathrm{ID}_{U_i} \| e \| KR_i) \cdot P + P_{\mathrm{pub}}) \stackrel{?}{=} g_1 \cdot g^D$ holds. Besides G is the message authentication code of the response message $\{C, E, F\}$ under the key $g_1 = \hat{e}(C, d_{S_j})$. The user can find out about any modification of the response message because the hash function $H_4(\cdot)$ is secure. Therefore, the proposed protocol can resist the modification attack.

(6) Replay attack: according to the proposed protocol, both two participants generate new random number $r_1, r_2 \in Z_q^*$ and $g_1 = g^{r_1}$, $g_2 = g^{r_2}$, which are involved in the login message and the response message. Due to the freshness of g_1, g_2, the user and the service provider can find the replay of messages by checking the validity of the received message. Therefore, the proposed protocol resists the replay attack.

(7) Man-in-the-middle attack: based on the above description, we conclude that the proposed protocol provides mutual authentication between two participants. Therefore, the proposed protocol can resist the man-in-the-middle attack.

5.4. Security Comparison. In this part, we compare the security of the proposed protocol with other multiserver architecture protocols in Table6. We introduce a new independent criterion, which is based on a widely adopted standard [40, 41] as follows:

C1 (no password verifier table): the server does not need to maintain a database for storing user passwords or some derived values of user passwords.

C2 (password friendly): the password is memorable and can be chosen freely and changed locally by the user.

C3 (no password exposure): the password cannot be derived by the privileged administrator of the server.

C4 (no smart card loss attack): the scheme is free from smart card loss attack, i.e., unauthorized users getting a victim's card should not be able to easily change the password of the smart card, recover the victim's password by using online, offline or hybrid guessing

Table 6: Performance comparison.

	Protocol rounds	Computation cost		Communication cost		Storage cost	Security requirements												
		User side	Server side	User side	Server side		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
Odelu et al. [14]	3	$T_{mtp} + 4T_h + 3T_{sm} + 2T_{\exp} \approx 78.519$	$T_{mtp} + 4T_h + 4T_{\exp} + T_{sm} + T_{\mathrm{mul}} + 2T_{bp} \approx 15.303$	832	672	1674	✓	✓	✓	×	×	✓	✓	✓	✓	✓	✓	✓	×
He et al. [15]	3	$2T_{sm} + T_{pa} + 2T_{\exp} + 8T_h \approx 31.837$	$T_{bp} + 4T_{\exp} + 2T_{\mathrm{mul}} + 5T_h \approx 5.246$	2240	1184	3424	✓	✓	✓	×	×	×	✓	✓	×	✓	✓	✓	×
Ours	2	$3T_{sm} + 2T_{\exp} + T_{pa} + 6T_h \approx 45.354$	$2T_{bp} + 4T_{\exp} + T_{sm} + T_{pa} + T_{\mathrm{mul}} + 7T_h \approx 11.107$	896	672	1834	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

attacks, or impersonate the user to login to the system, even if the smart card is obtained and/or secret data in the smart card are revealed.

C5 (resistance to known attacks): the scheme resists various kinds of basic/sophisticated attacks, including offline password guessing attack, replay attack, parallel session attack, desynchronization attack, stolen verifier attack, impersonation attack, key control, unknown key share attack, and known key attack.

C6 (sound repairability): the scheme provides smart card revocation with good repairability, i.e., a user can revoke her card without changing her identity.

C7 (provision of key agreement): the client and the server can establish a common session key for secure data communications during the authentication process.

C8 (no clock synchronization): the scheme is not prone to the problems of clock synchronization and time delay, i.e., the server needs not to synchronize its time clock with these time clocks of all input devices used by smart cards, and vice versa.

C9 (timely typo detection): the user will be timely notified if she inputs a wrong password by mistake when login.

C10 (mutual authentication): the user and server can verify the authenticity of each other.

C11 (user anonymity): the scheme can protect user identity and prevent user activities from being traced.

C12 (forward secrecy): the scheme provides the property of perfect forward secrecy.

C13 (hierarchical authentication): when a server authenticates with a user, it checks the user's authentication right. If the user authentication right belongs to the server authentication level, the authentication is successful. Otherwise, the authentication request is denied.

6. Performance Comparison

We show the computation and communication costs of the proposed protocol. We compare its performance with other protocol. For the purpose of getting a trusted security level (1024-bit RSA algorithm), an Ate pairing: $\widehat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$is used. $\mathbb{G}_1$ with order q is generated by a point on a supersingular elliptic curve $E(F_p)$: $y^2 = x^3 + 1$ which is defined on the finite field F_p. Order q is a 160-bit prime number and p is a 512-bit prime number.

6.1. Computation Cost Comparison. We give the running time of various operations performed in the proposed protocol, and we compare the results with He et al. [15] and Odelu et al. [14]. In this section, we use the following notations for the following running times in this paper:

(i) T_{bp}: the running time of a bilinear pairing operation

Table 7: The running time of related operations (milliseconds).

	T_{mtp}	T_{bp}	T_{sm}	T_{pa}	T_{exp}	T_{mul}	T_h
User	33.582	32.713	13.405	0.081	2.249	0.008	0.056
Server	4.174	1.665	1.665	0.011	0.260	0.001	0.006

(ii) T_{sm}: the running time of a scalar multiplication operation in $\mathbb{G}_1$

(iii) T_{mtp}: the running time of a map-to-point hash function in $\mathbb{G}_1$

(iv) T_{pa}: the running time of a point addition operation in $\mathbb{G}_1$

(v) T_{exp}: the running time of an exponentiation operation in $\mathbb{G}_2$

(vi) T_{mul}: the running time of a multiplication operation in $\mathbb{G}_2$

(vii) T_h: the running time of a general hash operation

The above operations are implemented with MIRACLE [42] library on a rooted android mobile phone (A5000 with Qualcomm 801 processor, 2-gigabyte memory, Google Android 4.4.2 operating system) and a personal computer (Lenovo with an i7-6700HQ 2.6 GHz, 16-gigabyte memory, Windows 7® operating system). The running time of those operations is listed in Table 7.

We compare the computation costs of the proposed protocol with He et al. [15] and Odelu et al. [14] based on the running time of the user and the service providerand the result is shown in Table 6.

6.2. Communication Cost Comparison. According to the description of the trusted security level, q is a 160-bit prime number and p is a 512-bit prime number. The size of an element in $\mathbb{G}_1, \mathbb{G}_2$ is 1024 bits. The size of the hash function's output is 160 bits, and the identity and the expire parameter are both 32 bits. In our protocol, we only have two rounds of communication for establishing a session key. On client side, the messages C, E, F require 320 + 320+256 = 896 bits, and on service provider side, messages G, g_2 require 160 + 512 = 672 bits. The total communication costs are 1568 bits. In He et al.'s [15] protocol, on client side, messages R_{U_i}, C_{U_i} require 1024 + 32+1024 + 160 = 2240 bits, and on server side, messages y, α_{S_j} require 1024 + 160 = 1184 bits. In Odelu et al.'s [14] protocol, on client side, messages M_1, M_3 require 320 + 512 = 832 bits, and on server side, message M_2 require 672 bits. The comparison of communication costs is shown in Table 6.

6.3. Storage Cost Analysis. Because the mobile devices are limited to storage spaces, we therefore analyze the storage cost on the user side to show the proposed protocol has reasonable storage cost. In Odelu et al.'s protocol, a user needs to store $\{E_{i,Lt_i}, e_i, \theta_i, t, e, Lt_i, P, P_{\text{pub}}, g, q\}$ in his/her device, which costs 1674 bits. In He et al.'s protocol, user needs to store $\{R_{u_i}, y, a_{s_j}, g_{U_i}, \psi_{U_i}, v_{U_i}, b_{U_i}\}$, which costs 3230

bits. In our protocol, user needs to store $\{A, B, \theta_i, a_i, e, P, P_{\text{pub}}, g, q\}$, which costs 1834 bits.

7. Conclusion

In this paper, we have proposed a hierarchical authentication protocol for the multiserver environment. The significant contribution of this paper is that we have built an authentication right tree based on the Merkle hash tree, which can be used to verify the authentication right of a user when he/she is authenticating with the service provider. The extended hierarchical authentication feature has added more flexibility and security to multiserver architecture. The security proof has demonstrated that our protocol is provably secure under the random oracle model. Our protocol has reasonable computation and communication costs, which could be suitable for multiserver architecture.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was funded by the Chengdu Science and Technology Bureau (no. 2016-XT00-00015-GX) and the Civil Aviation Administration of China (no. PSDSA201802).

References

[1] H. Li, Y. Dai, L. Tian, and H. Yang, "Identity-based authentication for cloud computing," in *Cloud Computing*, M. G. Jaatun, G. Zhao, and C. Rong, Eds., Springer, Berlin, Germany, pp. 157–166, 2009.

[2] H. Ning, H. Liu, and L. T. Yang, "Aggregated-proof based hierarchical authentication scheme for the internet of things," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 657–667, 2015.

[3] H. Tohidi and V. T. Vakili, "Lightweight authentication scheme for smart grid using merkle hash tree and lossless compression hybrid method," *IET Communications*, vol. 12, no. 19, pp. 2478–2484, 2018.

[4] M.-H. Shao and Y.-C. Chin, "A privacy-preserving dynamic id-based remote user authentication scheme with access control for multi-server environment," *IEICE Transactions on Information and Systems*, vol. E95-D, no. 1, pp. 161–168, 2012.

[5] D. He and D. Wang, "Robust biometrics-based authentication scheme for multiserver environment," *IEEE Systems Journal*, vol. 9, no. 3, pp. 816–823, SEP 2015.

[6] V. Odelu, A. K. Das, and A. Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1953–1966, 2015.

[7] Q. Xie, D. S. Wong, G. Wang, X. Tan, K. Chen, and L. Fang, "Provably secure dynamic id-based anonymous two-factor authenticated key exchange protocol with extended security model," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1382–1392, 2017.

[8] P. Chandrakar and H. Om, "A secure and robust anonymous three-factor remote user authentication scheme for multi-server environment using ecc," *Computer Communications*, vol. 110, pp. 26–34, 2017.

[9] Q. Feng, D. He, S. Zeadally, and H. Wang, "Anonymous biometrics-based authentication scheme with key distribution for mobile multi-server environment," *Future Generation Computer Systems*, vol. 84, pp. 239–251, 2018.

[10] R. Amin, N. Kumar, G. P. Biswas, R. Iqbal, and V. Chang, "A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment," *Future Generation Computer Systems*, vol. 78, pp. 1005–1019, 2018.

[11] J. Cui, X. Zhang, N. Cao, D. Zhang, J. Ding, and G. Li, "An improved authentication protocol-based dynamic identity for multi-server environments," *International Journal of Distributed Sensor Networks*, vol. 14, no. 5, 2018.

[12] K. Choi, J. Hwang, D. Lee, and I. Seo, "Dased authenticated key agreement for low-wer mobile devices," C. Boyd and J. M. Gonzalez Nieto, Eds., in *Proceedings of the10th Australasian Conference on Information Security and Privacy*, vol. 3574, pp. 494–505, Brisbane, Australia, July 2005.

[13] Y.-M. Tseng, S.-S. Huang, T.-T. Tsai, and J.-H. Ke, "List-free ID-based mutual authentication and key agreement protocol for multiserver architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 1, pp. 102–112, 2016.

[14] V. Odelu, A. K. Das, S. Kumari, X. Huang, and M. Wazid, "Provably secure authenticated key agreement scheme for distributed mobile cloud computing services," *Future Generation Computer Systems*, vol. 68, pp. 74–88, 2017.

[15] D. He, S. Zeadally, N. Kumar, and W. Wu, "Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2052–2064, 2016.

[16] A. Irshad, M. Sher, H. F. Ahmad, B. A. Alzahrani, S. A. Chaudhry, and R. Kumar, "An improved multi-server authentication scheme for distributed mobile cloud computing services," *KSII Transactions on Internet and Information Systems*, vol. 10, pp. 5529–5552, 2016.

[17] J.-L. Tsai and N.-W. Lo, "A privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, vol. 9, no. 3, pp. 805–815, 2015.

[18] L. Xiong, D. Peng, T. Peng, and H. Liang, "An enhanced privacy-aware authentication scheme for distributed mobile cloud computing services," *KsII Transactions on Internet and Information Systems*, vol. 11, no. 12, pp. 6169–6187, 2017.

[19] L. Xiong, D. Y. Peng, T. Peng, H. B. Liang, and Z. C. Liu, "A lightweight anonymous authentication protocol with perfect forward secrecy for wireless sensor networks," *Sensors*, vol. 17, no. 11, p. 2681, 2017.

[20] S. Kumari, A. K. Das, X. Li et al., "A provably secure biometrics-based authenticated key agreement scheme for multi-server environments," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2359–2389, 2018.

[21] G. Xu, S. Qiu, H. Ahmad et al., "A multi-server two-factor authentication scheme with un-traceability using elliptic curve cryptography," *Sensors*, vol. 18, no. 7, p. 2394, 2018.

[22] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, vol. 12, no. 2, pp. 2039–2042, 2018.

[23] S. Chatterjee, S. Roy, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos, "Secure biometric-based authentication scheme using Chebyshev chaotic map for multi-server environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 824–839, 2018.

[24] W.-S. Juang, "Efficient multi-server password authenticated key agreement using smart cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 251–255, 2004.

[25] S. Barman, A. K. Das, D. Samanta, S. Chattopadhyay, J. J. P. C. Rodrigues, and Y. Park, "Provably secure multi-server authentication protocol using fuzzy commitment," *IEEE Access*, vol. 6, pp. 38578–38594, 2018.

[26] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: how to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology—EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds., Springer, Berlin, Germany, pp. 523–540, 2004.

[27] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2015.

[28] P. Wang, Z. Zhang, and D. Wang, "Revisiting anonymous two-factor authentication schemes for multi-server environment," in *Proceedings of the 2018 International Conference on Information and Communications Security*, Lille, France, October 2018.

[29] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology—CRYPTO '87*, C. Pomerance, Ed., Springer, Berlin, Germany, pp. 369–378, 1988.

[30] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2009, https://metzdowd.com.

[31] A. G. Reddy, E.-J. Yoon, A. K. Das, V. Odelu, and K.-Y. Yoo, "Design of mutually authenticated key agreement protocol resistant to impersonation attacks for multi-server environment," *IEEE Access*, vol. 5, pp. 3622–3639, 2017.

[32] S. Jangirala, S. Mukhopadhyay, and A. K. Das, "A multi-server environment with secure and efficient remote user authentication scheme based on dynamic ID using smart cards," *Wireless Personal Communications*, vol. 95, no. 3, pp. 2735–2767, 2017.

[33] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract)," in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing—STOC '98*, pp. 419–428, Dallas, TX, USA, May 1998.

[34] C. Ran and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, Amsterdam, Netherlands, April 2002.

[35] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[36] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptology—EUROCRYPT 2001*, B. Pfitzmann, Ed., Springer, Berlin, Germany, pp. 453–474, 2001.

[37] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[38] S. Kumari, X. Li, F. Wu, A. K. Das, K.-K. R. Choo, and J. Shen, "Design of a provably secure biometrics-based multi-cloud-server authentication scheme," *Future Generation Computer Systems*, vol. 68, pp. 320–330, 2017.

[39] A. G. Reddy, A. K. Das, V. Odelu, and K.-Y. Yoo, "An enhanced biometric based authentication with key-agreement protocol for multi-server architecture based on elliptic curve cryptography," *PLoS One*, vol. 11, no. 5, Article ID e0154308, 2016.

[40] D. Wang and P. Wang, "Two birds with one stone: two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2018.

[41] D. Wang, W. Li, and P. Wang, "Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4081–4092, 2018.

[42] S. S. Ltd, "BWorld robot control software," 2015, http://www.shamus.ie/index.php?page=home.