Hindawi
Mobile Information Systems
Volume 2020, Article ID 2964562, 16 pages
https://doi.org/10.1155/2020/2964562

Research Article

Smart Contract-Based Cross-Domain Authentication and Key Agreement System for Heterogeneous Wireless Networks

Guangsong Li,[1,2] Yang Wang ,[1] Bin Zhang,[1] and Siqi Lu[1]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[2]*Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China*

Correspondence should be addressed to Yang Wang; 275119624@qq.com

Received 14 February 2020; Revised 24 August 2020; Accepted 27 August 2020; Published 8 September 2020

Academic Editor: Floriano Scioscia

Nowadays, it is still a major challenge to design a secure cross-domain authentication protocol for heterogeneous wireless networks with different security parameters. As a new technology, blockchain has attracted people's attention because of its tamper-proof and decentralized characteristics. In this paper, we propose a cross-domain authentication and key agreement system based on smart contract of blockchains. Public keys of the nodes are managed using the smart contracts, and the system parameters are confirmed by contract query. On this basis, a cross-domain authentication and key agreement protocol is designed. In this protocol, roaming users can select temporary authentication parameters according to the system parameters of the roaming domain to complete authentication and key agreement, and users are anonymous in the process. Security of the protocol is demonstrated under the CK model, and two formal analysis tools are used to further analyze the protocol. Since the protocol does not have complex cryptographic operations and certificate verification, it has lower computational and communication overhead.

1. Introduction

With the development of the Internet and the increase of wireless access devices such as smartphones, laptops, and iPads, people demand more network resources and better network services. Various wireless access technologies have been developed and deployed to meet growing demand, such as CDMA (Code Division Multiple Access), Wi-Fi (Wireless Fidelity), Wi-MAX (Worldwide Interoperability for Microwave Access), and LTE (Long Term Evolution). These technologies have their advantages and disadvantages, and no wireless access technology is perfect to meet the needs of all users. In this context, heterogeneous wireless networks that incorporate multiple access technologies have emerged to take full advantage of the network characteristics of various access technologies. The upcoming 5G [1] and the Internet of Things (IoT) [2] has a typical heterogeneous structure. In 5G, multiple wireless access technologies coexist, and macro stations responsible for wide-area coverage and low-power small stations responsible for hotspot coverage are developed in multilayer.

The purpose of heterogeneous wireless network convergence is to give full play to the advantages of various wireless network resources, so that users can select a suitable access network according to their needs. Users in a heterogeneous network can choose to access or handover to a wireless network that best suits their needs according to current network status and service requirements. Multiple independent and autonomous security domains in heterogeneous wireless networks typically have different security standards, and each domain uses different system parameters. Therefore, a cross-domain authentication key agreement solution that does not restrict domain system parameters is required.

Traditionally, in cross-domain authentication solutions, there are two main frameworks. One is based on the symmetric key scheme such as Mahshid and Eslamipoor [3]. Although the authentication protocol based on symmetric key is low in complexity and easy to implement, the burden of generating, distributing, storing, and managing shared keys is complex and huge. Especially for heterogeneous networks, it will increase the complexity of system

management and reduce the scalability. The other is based on the traditional certificate, which has the burden of certificate management and distribution and results in high computation and communication overhead. Millán et al. [4] adopted the Certificate Authority (CA) scheme to establish a bridge CA model that all domains trust. This scheme requires all domains to trust this trusted third party, which is difficult to apply in practice, and there is also the problem of how to obtain certificate status information across domains.

In addition, identity-based cryptography is used to facilitate cross-domain authentication. Peng [5] proposed a multidomain authentication key agreement protocol based on the identity cryptography. The protocol requires all authentication servers to be trusted, and each authentication server uses the same PKG (Private Key Generator) system parameters, which makes the system poor scalability. Papers [6, 7], respectively, gave cross-domain authentication protocols based on identity proxy signatures, which require the agent to establish a security association with the trust domain. But, the signature authorization from the original signer to the proxy signer may bring more security risks, and the introduction of proxy mechanisms increases system complexity. In conclusion, there is a need for a common PKG in most of the current cross-domain authentication key agreement schemes using identity-based cryptography. In 2003, Chen et al. [8] first proposed a user key agreement protocol under different PKGs. In 2004, McCullagh and Barreto [9] proposed a key agreement protocol with key escrow and unmanaged modes in different PKG environments, but then the protocol pointed out that it could not resist key leakage attacks. In recent years, some identity-based key agreement schemes [10, 11] and certificateless authentication key agreement schemes [12] have been proposed one after another, but they cannot meet the requirements of key agreement between different trust domains. However, in the future heterogeneous wireless network application, each trust domain is mostly an independent autonomous domain, where different system parameters are used. Therefore, most of the above solutions are difficult to meet the authentication and key agreement requirements of the heterogeneous wireless network.

Some anonymous cross-domain schemes have been proposed one after another. In 2014, Cheng et al. [13] proposed a distributed anonymous authentication (DAA) protocol, which uses an unlinkable group signature algorithm to provide authentication without sharing keys in advance, which significantly reduced signaling overhead while protecting privacy. In 2017, Fu et al. [14] proposed a scheme based on the (t, n) shared secret key to protect the privacy of users during the handover process and use the unpaired identity encryption method to achieve highly efficient handover. In 2018, a novel group key management protocol [15] for cross-domain dynamic anonymous authentication was proposed to realize cross-domain secure anonymous group communication. However, the above schemes also have the problem of using the same parameters in different domains.

Recently, some other authentication methods were proposed for different network applications. Lu et al. presented an anonymous three-factor key agreement using Elliptic Curve Cryptography (ECC), which is for secure communications to be used in resource-constrained wireless sensor networks [16]. Cheng et al. propose a novel design using an asymmetric bivariate polynomial for user authentication and group key establishment with low communication costs in WSNs [17]. Arezou et al. propose a secure and lightweight authentication and key agreement protocol for IoT based WSNs that concerns the strong replay attacks and perfect forward secrecy [18]. To ensure secure communication over the insecure public network, Qi and Chen propose a privacy-preserving biometrics-based authenticated key agreement scheme using ECC, which has perfect user experience in changing password without interacting with the server [19]. Akram et al. propose an anonymous multiserver authentication which allows for getting services from different servers using only single-time registration [20].

In 2008, Nakamoto designed the Bitcoin system and introduced blockchain technology for the first time in his paper [21]. Blockchain is a distributed ledger technology and a decentralized storage system. In 2014, the blockchain technology began to be applied to distributed applications by introducing smart contract. In 2014, based on the Bitcoin blockchain system, Fromknecht proposed the first distributed PKI authentication system, Certcoin [22, 23]. Certcoin is used instead of CA to provide efficient key query and identity retention. But, it has the problem of user privacy leakage because the binds of user identities and public keys are directly recorded in the public ledger of the blockchain. Axon proposed an improved Certcoin scheme [24], which was a PKI privacy protection authentication system. In 2016, Lewison proposed a certificate-based PKI authentication system using the Ethereum platform [25], which solved the problem of excessive traffic of the traditional PKI certificate management and the use of certificate revocation list (CRL) and online certificate status protocol (OCSP). We refer to these existing schemes to design distributed PKI for wireless networks based on smart contracts. Wang et al. [26] proposed a blockchain-based cross-domain authentication model named BlockCAM to enable users to access shared resources across domains in a secure way. But when the number of nodes is large, its authentication efficiency is low because of the need for traversing the blockchain. Besides, the scheme has not referred to the key agreement. The comparison of these related protocols mentioned above is provided in Table 1. Unfortunately, there is no blockchain-based solution to solve the cross-domain authentication problem of heterogeneous wireless networks so far.

1.1. Contribution and Motivation. Since the existing schemes are centralized and vulnerable to single point failures and denial of service attacks, they are unsuitable for heterogeneous environments for using different parameters in multidomain. Based on the decentralized and distributed blockchain and the distributed and easy-to-program smart contracts, we propose a smart contract-based cross-domain authentication and key agreement system for heterogeneous

Table 1: Comparison of some related protocols.

Protocols in references	Technology	Strengths	Weaknesses
[3]	Symmetric key	Low in complexity and easy to implement	High complexity of system management and low scalability
[4]	Bridge CA	Flexible authentication	High computation and communication overhead; requiring trusted third party
[5] [6, 7]	Identity-based cryptography Identity proxy signatures	Without public key certification	The same PKG; low scalability
[8] [9] [10, 11]	Key agreement Paring free key agreement	Different PKG; without public key certification	High computation overhead
[12]	Certificateless authentication key agreement	Without public key certification	Complex cryptographic operations
[13] [14] [15]	Group signature algorithm (t, n) shared secret key Group key management	Anonymous cross-domain authentication	Complex cryptographic operations
[16, 19, 20]	ECC	Anonymous multifactor authentication	Complex cryptographic operations
[17]	Asymmetric bivariate polynomial	Low communication cost	
[18]	Biohashing function	Multifactor and lightweight authentication	High communication cost
[22, 23]	Blockchain	Efficient key management	User privacy leakage
[24]		Privacy protection	
[25]		Low communication cost	High computation overhead
[26]		Cross-domain authentication	

wireless networks. The system constructs a blockchain network in which the CA and access point (AP) nodes of each domain are set as blockchain nodes, and the public key of AP nodes and the hash of the public key of registered users are recorded in smart contracts by CA nodes. Cross-domain authentication is realized by mutual query and verification of the public keys stored in the contract instead of the traditional PKI method with mutual issuance of signed certificates and verification of signatures. Our solution implements cross-domain authentication between domains with different parameters and guarantees user anonymity. Evaluation results show that the solution has low communication overhead and computation cost.

1.2. Organization. This paper is organized as follows: Section 2 describes the blockchain system and the CK model. In Section 3, we describe our proposed scheme in detail. In Section 4, we give its security proof under the CK model and other security analysis and results of formal analysis tools. Section 5 shows the real implementation and evaluation results. The paper is concluded in Section 6.

2. Preliminaries

In this section, we will introduce blockchain and CK model in provable security theory briefly, where blockchain improves the security and effectiveness of our system and the CK model helps us to analyze protocol security.

2.1. Blockchain and Smart Contract. Blockchain is a kind of decentralized ledger running on the p2p network that combines data blocks into a specific data structure in the form of chains in the chronological order. The blockchain mainly has three characteristics, namely, distributed multicenter, collective maintenance, and tamper-resistant. The characteristics of the blockchain make it a useful technology for building distributed and transparent storage systems where records cannot be hidden or destroyed by third parties.

Blockchain can be divided into two categories [27]: authorized and unauthorized. The unauthorized blockchain is public blockchain like Bitcoin and Ethereum. It is a blockchain that is open to all and anyone can participate. It usually consumes a lot of energy and time because it involves computational efforts to enhance system security against modification attacks. And, the authorized blockchain is private or consortium blockchain such as Hyperledger Fabric [28]. It limits the consensus peers (only selected trust peers named as committing peers have the right to verify the transaction and generate a new block). It is neither energy-consuming nor time-consuming. Partial decentralization, better permission management, and privacy protection of the consortium blockchain make it better for enterprises and specific scenarios.

Currently, designing programmable currencies and contracts have become a trend to extend blockchain applications beyond the cryptocurrency field. Smart contracts are ways to use blockchains to implement agreements between parties rather than relying on third parties to maintain a trust relationship. Smart contracts are responsible for

implementing, compiling, and deploying the business logic of blockchain system in the form of code, triggering the automatic execution of established rules and minimizing manual intervention. Smart contracts allow both parties to participate and can partially or fully execute or enforce certain commitments or agreements, which are a set of commitments in the form of digital [29]. A smart contract is essentially a collection of predefined instructions and data that have been recorded at a specific address in the blockchain. By encapsulating operational logic into bytecode and performing Turing complete computations for distributed miners, smart contracts allow users to transcode more complex business models into new transactions on blockchain networks. Smart contracts can be programmed using the Turing Complete Language. The Turing Complete Language is a programming language that assumes that any computational problem can be solved with sufficient time and space. Typically, smart contracts are compiled into a specific binary format and deployed by the account to a global database of blockchains. Smart contracts provide a promising solution for implementing a more flexible and convenient public key management model on a blockchain network.

2.2. Provable Security Theory

2.2.1. CDH Assumption. Let G be a cyclic addition group and $P \in G$ be a generator of order q; given P, aP, bP for random $a, b \in Z_q^*$, it is difficult to calculate abP.

2.2.2. CK Security Model. Canetti and Krawczyk [30] extended the model of the paper [31] and proposed the CanettiKrawczyk (CK) model. The CK model defines security with indistinguishability. If the attacker cannot distinguish between the session key generated by the protocol and an independent random value under its allowed attack capability, the key agreement protocol is secure. The CK model defines the session key secure (SK-secure) and presents a modular approach to demonstrating protocol security using SK-secure definitions.

The CK model consists of three parts: an authenticated-link adversarial model (AM), an unauthenticated-link adversarial model (UM), and an authenticator. The authenticator is the link between the AM and the UM. The AM model is an authenticated link adversarial model in an ideal environment. The attacker is passive in AM and cannot forge, tamper with, or replay messages from uncaptured participants. And, it is restricted to faithfully deliver the same message once (although the order of delivery can be delayed or rearranged). In addition, the attacker can also perform the following attacks: party corruption, session-key query, session state reveal, and test-session query.

Definition 1. Test-session query: an attacker can select a test session from those completed, unexpired, and unexposed sessions at any time during the protocol run to obtain a test-session key or a random number. Specifically, let sk be the session key of the test session. When the attacker queries the test session, a coin b is tossed. If $b = 0$, sk is returned to the attacker; otherwise, a value r randomly chosen from the probability distribution of keys is returned to the attacker. Finally, the attacker outputs b' as its guess for b.

The UM model is an unauthenticated links adversarial model in a real network environment. In addition to executing all the attacks in AM mentioned above, the attacker can also completely control the network, including inserting, replaying, forging, and tampering with messages. In UM, the attacker can control the scheduling of protocol events and communication links. At the same time, the attacker can also know the secret information of the protocol participant through specific attack means.

Definition 2 (SK-secure). For any adversary $\mathcal{U}$ in the UM, a protocol is SK-secure if the following properties hold:

(1) After two uncorrupted parties complete matching sessions, they both output the same session key.

(2) The adversary $\mathcal{U}$ initiates a test-session query attack and the probability that $\mathcal{U}$ guesses correctly the bit b is no more than 1/2 plus a negligible fraction in the security parameter.

Theorem 1 (see [30]). *Suppose λ is a message transmission (MT) authenticator, that is, λ emulates a simple MT protocol in UM. Suppose C_λ is a compiler constructed based on λ, then C_λ is also an authenticator. The authenticator is a very important mechanism in the modular approach, which ensures that the security protocols in the AM are translated into security protocols in UM.*

The proof of Theorem 1 is detailed in the paper [30]. The papers [30–32] detail the basic theory of the CK security model and the basic method of designing a secure key agreement protocol based on the model. For more detailed information about CK model and its application, refer [33–35].

3. Smart Contract-Based Cross-Domain Authentication and Key Agreement System

In order to provide continuous services for mobile users securely, it is necessary to design a secure and efficient cross-domain authentication protocol for wireless networks. Blockchain is one of promising techniques for next-generation wireless networks, which may establish a secure and decentralized resource sharing environment. Once recorded, the data on the blockchain cannot be tampered with. Currently, many blockchain-based schemes as Section 1 have been proposed and leveraged to enhance security. Moreover, a decentralized, trusted, and publicly auditable database could be built based on blockchain in wireless networks, so that decentralized trust can be achieved. Using the decentralized blockchain and the easy-to-program smart contracts, we propose a smart contract-based cross-domain authentication and key agreement system for heterogeneous wireless networks.

3.1. System Model.

As we introduce in Section 2.1, the blockchain is a tamper-proof, antiforgery, and distributed storage system and the smart contract is distributed,

traceable, and persistently running. Based on these characteristics, a smart contract-based authentication and key agreement system is designed for heterogeneous wireless networks. For the system, based on the needs of the actual network environment, our cross-domain authentication and key agreement system should meet the following basic security requirements [36].

3.1.1. Single Registration. For practice, all nodes in the system can authenticate or communicate with other registered nodes only if they are registered only once.

3.1.2. User Anonymity. The system should ensure that the user ID is not visible to attackers and the AP nodes to protect the anonymity of the user node.

3.1.3. Mutual Authentication. Nodes in the system can believe each other's identity, ensure that the identity claimed by the other party is itself, and confirm that the message is from the real sender.

3.1.4. Session Key Agreement. To communicate securely between nodes, the system should negotiate a session key with another party during the authentication phase for subsequent communication.

3.1.5. Perfect Forward Secrecy. To prevent the leakage of the session key of the previous communication and protect the previous communication content, any attacker cannot recover the previous session key even if he obtains the private key of both communication parties.

3.1.6. No Online Certificate Authority. To reduce the communication cost, the system should avoid online certificate authorities participating in the authentication and any two nodes can directly authenticate each other without relying on an online certificate authority.

3.1.7. Resilience to Common Attacks. The system should be designed to resist common attacks, such as impersonation attacks, modification attacks, replay attacks, man-in-the-middle attacks, and denial of service attacks or distributed denial of service attacks (DoS/DDoS).

Note: the property of "no certificate authority" is very important for system security. The readers can refer to [33–35] for more information about it.

The system includes a smart contract-based public keys management system (SCPKM) and a cross-domain authentication and key agreement protocol (CAKA). As described in Section 2.1, consortium blockchain, which is partially decentralized, can reach a consensus more quickly and give different privileges to different nodes. In our system, APs have certain computing and storage capacity as general nodes of blockchain for querying and invoking function in contract, and CAs have sufficient computing and storage capacity to complete the consensus task as committing peers of blockchain. A consortium blockchain is built on all AP nodes and CA nodes.

The system consists of APs, CAs, users, blockchain network, and smart contracts, as shown in Figure 1. There are several security domains (two domains in the figure for simplicity), and each security domain consists of one CA, several APs, and many users.

The two protocols SCPKM and CAKA are described in detail below.

3.2. Domain Initialization. There exist some APs and a CA (for simplicity, only one CA is set; in fact, a certain number of CAs should be set according to the size of the domain) in each security domain. Each CA chooses independently different or same (according to security requirement of the domain) system public parameters. We take domain U as an example to illustrate the generation of system public parameters in the domain as follows. A large prime p_U is selected, E_U is an elliptic curve defined on a finite field F_{p_U}, and P_U is a generator of E_U. H^U is a cryptography hash function, where H^U: $E_U \longrightarrow \{0,1\}^*$. The basic public parameter basicpare^U is $<p_U, E_U, P_U, H^U>$. Define a key generation algorithm Gen_U: $\text{basicpare}^U \longrightarrow (A^U, a^U)$, where a^U is randomly chosen in $Z^*_{p_U}$ and $A^U = a^U \cdot P_U$. The CA generates public and private key pair (s_U, PK_U) using Gen_U. The system public parameter pare^U of the domain U is $<p_U, E_U, P_U, H^U, \text{Gen}_U, \text{PK}_U>$.

The CA and APs in a certain domain join the consortium blockchain as an organization.

3.3. Smart Contract-Based Public Key Management System. SCPKM is a protocol for the CA managing public keys. It achieves the decentralization storage of public keys. And, nodes such as users and APs could quickly query other node public keys to verify the node identity. Only if users have registered to a CA, they can be authenticated by any AP. It manages public keys of all nodes during the node registration, the public key revocation, and the public key update. We define some notations in Table 2 to describe the scheme clearly. In addition, CA will issue an authentication ticket to each registered user for anonymity:

$$\text{authentication ticket: } \cdot\{\text{PK}, \text{DI}_{\text{user}}, T_{\text{start}}, T_{\text{end}}, \text{sig}\}, \quad (1)$$

where PK is the public key of user, DI_{user} is the ID of user's domain, T_{start} and T_{end} are the ticket authorization effective and expiration time, and sig is a signature of $\{\text{PK}, \text{DI}_{\text{user}}, T_{\text{start}}, T_{\text{end}}\}$.

3.3.1. Contract Deployment. Algorithm 1 is shown in Table 3. The only CA in each domain compiles and deploys the smart contract to manage APs, users, and their public keys of its own domain. At deployment time, the function PK_domain in the contract will be invoked automatically, and information of the domain is written to the contract. Once smart contracts pass the validation process, they will be

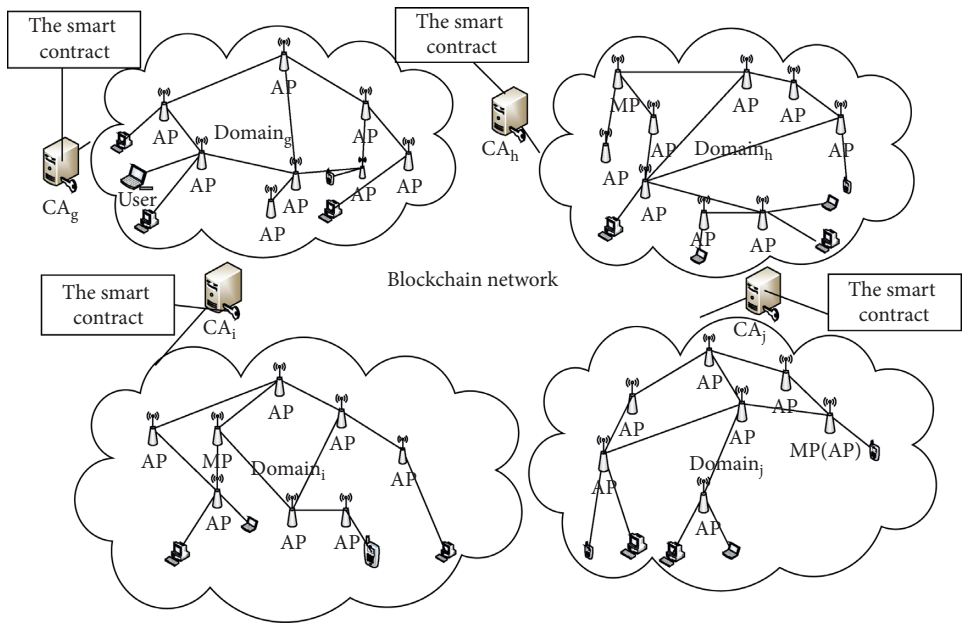

Figure 1: Heterogeneous wireless networks with blockchain.

Table 2: Constructor of the smart contract.

Notations	Description
H: $(\cdot)$	A cryptography hash function H: $\{0,1\}^* \longrightarrow \{0,1\}^l$ and l is a fixed constant.
sig (sk, m)	A signature algorithm that signs a message m using the private key sk
ver (PK, σ, m)	A verification function that verifies whether σ is a valid signature on m under the public key PK
$A \longrightarrow B$: m	Node A sends a message m to node B
DI_*, ID_*	A domain identifier and a node identifier
$E_k(m)$	A symmetric encryption algorithm that encrypts m with key k
s_N^U, PK_N^U	The private key and the public key of a node generated by using Gen_U of the domain U
CA_U	The CA of a node U
σ_*	Signature value

recorded in the blockchain forever by all the blockchain peers (APs and CAs). Then, all nodes can query the variable CA to get information of the domain.

3.3.2. Node Registration

(1) AP Registration. At the point of submission, the authors may provide all figures embedded within the manuscript at a convenient break near Suppose an AP node AP in domain U will register to its CA_U. It will generate its public and private key pair and store the public key to the smart contract, and then CA_U signs its ID and public key to confirm the registration. The registration process is as follows (related functions are described in Table 3 (Algorithm 1) and Table 4 (Algorithm 2)):

Step 1. AP sends CA_U the registration request $\{\mathrm{ID}_{\mathrm{AP}}, \mathrm{identification} - \mathrm{information}\}$ through a secure channel.

Step 2. After CA_U receives the registration request, and it verifies the identification-information and checks if $\mathrm{ID}_{\mathrm{AP}}$ has registered. If verification is correct, CA_U adds AP into the organization in the blockchain.

Step 3. Upon addition into the blockchain, AP invokes the function get_CA() to get the basic public parameter $\mathrm{basicpare}^U$ and the public key PK_U of the domain U, and then it generates its public and private key pair $(\mathrm{PK}_{\mathrm{AP}}^U, s_{\mathrm{AP}}^U)$ using Gen_U. Finally, it invokes AP _ Register $(\mathrm{ID}_{\mathrm{AP}}, \mathrm{PK}_{\mathrm{AP}}^U)$. AP computes the signature $\sigma_1 = \mathrm{sig}(s_{\mathrm{AP}}^U, m_1 = H(\mathrm{ID}_{\mathrm{AP}} \| \mathrm{identification} - \mathrm{information}))$ and then it sends CA_U the message $M_1 = \{\mathrm{ID}_{\mathrm{AP}}, \sigma_1\}$.

The signature demonstrates that the AP has the corresponding private key s_{AP}^U.

Step 4. When CA_U receives M_1, it verifies the signature σ_1 by calculating ver $(\mathrm{PK}_{\mathrm{AP}}^U, \sigma_1, m_1)$. If the verification is correct, CA_U computes the signature $\sigma_2 = \mathrm{sig}(s_U, H(\mathrm{ID}_{\mathrm{AP}} \| H^U(\mathrm{PK}_{\mathrm{AP}}^U) \| t_{\mathrm{begin}} \| t_{\mathrm{end}}))$ and then invokes the function CA _ Confirm $(\mathrm{ID}_{\mathrm{AP}}, t_{\mathrm{begin}}, t_{\mathrm{end}}, \sigma_2)$ to confirm the registration of AP. Otherwise, CA_U sends 'ERROR' and the error reason to AP.

(2) User Registration. Suppose a user UE in domain U will register to its CA_U. It will generate its master public and private key pair and then sends the public key, identification-information, and its signature to CA_U through a secure

Table 3: Constructor of the smart contract.

Algorithm 1 constructor of smart contract
Structure AP_PK % define the structure of information of AP's public key. ID; % ID of an AP. PK; % the public key of the AP. T_begin; % the public key effective start time. T_end; % the public key expiration time. sig; % CA's signature of ID, PK, T_begin, and T_end. Structure CA_Info CA % define the structure of information of CA. And CA, an CA_Info structure variable is public to all node ID_domain; % domain ID Basicpare; % the basic public parameter of the domain. PK; % the public key of CA. sig; % CA's signature of ID_domain and PK. Map(hash- > bool) user % A map denote if user is registered it returns true. Map(ID- > AP_PK) ap % A map denote if ID is registered it returns the public key information of the node corresponding to ID. Uint len; % the number of nodes. function PK_domain (id, pare, pk, sig) % constructor, it is automatically invoked when this smart contract is deployed; initialize CA and only this function can modify variable CA. owner = sender.addr; % Define CA is the owner of the contract. CA = {id, pare, pk, sig}; len = 0; function get_CA(·) % Invoked to obtain the information of the domain. return CA;

Table 4: Register to the smart contract.

Algorithm 2 register

function AP_Register (id, pk)
% AP node invokes the function to register to CA.
If(AP_node[id] = NULL) % id has not been registered.
AP_node[id] = sender.addr; len ++;
ap[id] = = { id, pk, NUL, NUL, NUL};
function CA_Confirm(id, *t*_begin, *t*_end, sig)
% CA invokes the function to confirm the registration of AP by writing the *t*_begin, *t*_end and signature sig to the contract.
if(sender.pk = = owner) % Guarantee only CA can invoke the function.
ap[id]. T_begin = *t*_begin;
ap[id]. T_end = *t*_end;
ap[id].sig = sig;
function user_Register (hash)
% CA invokes the function to write the hash of user public key to the contract.
if(sender.addr = = owner) % Guarantee only CA can invoke the function.
User[hash] = true;
function getAP(id)
% invoked to obtain the public key of an AP.
return ap[id];
function user_Verificate (hash)
% Invoked by someone to check a certain hash is registered or not.
if(user[hash])
return true;
else return false;

channel. CA_U stores the hash of the public key to the smart contract and sends the authentication ticket to user through the secure channel. The registration process is as follows (related functions are listed in Table 3 (Algorithm 1) and Table 4 (Algorithm 2)).

Step 1. UE sends CA_U the registration request $\{ID_{UE}, \text{identification} - \text{information}\}$ through a secure channel.

Step 2. CA_U verifies the identification-information and checks if ID_{UE} has registered. If UE passes further verification, CA_U gives UE permission to query the blockchain.

Step 3. UE invokes the function get_CA( ) to get the basic public parameter $basicpare^U$ and the public key PK_U of the domain U, and then it generates its public and private key pair (PK_{AP}^U, s_{AP}^U) using Gen_U. UE computes the signature $\sigma_1 = \text{sig}(s_{AP}^U, m_1 = H(ID_{UE} \| \text{identification} - \text{information}))$ and sends message $M_2 = \{ID_{UE}, PK_{UE}, \sigma_1\}$ to CA_U through the secure channel.

Step 4. Upon receiving M_2, CA_U verifies the signature σ_1 by calculating $\text{ver}(PK_{UE}, \sigma_1, m_1)$. If the verification is wrong, CA_U sends "ERROR" and the error reason to UE. Otherwise, CA_U computes $h = H(H^U(PK_{UE}^U))$, invokes function user_Register(h), and maintains the user list (ID_{UE}, PK_{UE}^U, h). CA_U computes $\sigma_2 = \text{sig}(s_U, H(H^U(PK_{UE}^U) \| DI_U \| t_{\text{begin}} \| t_{\text{end}}))$ and generates the authenticate ticket $T_{UE}^U = \{PK_{UE}^U, DI_U, t_{\text{begin}}, t_{\text{end}}, \sigma_2\}$. Finally, CA_U sends T_{UE}^U to through the secure channel.

3.3.3. Public Key Update. When the user or AP needs to update its public key, it generates a new key pair and sends the new public key to CA through the secure channels. CA invokes user_update or AP_Update function in Table 5 (Algorithm 3) to update public key in the contract and then CA updates the corresponding user list for user update.

3.3.4. Public Key Revocation. When an AP or a user detects some node's suspicious behavior or it detects some node A is broken, it reports the abnormal case to CA. Then the CA checks the report. If it is true, the CA invokes AP_Revoke or user_Revoke in Table 6 (Algorithm 4) to revoke public key of the node A and other nodes will refuse communication with the node A.

3.4. Cross-Domain Authentication and Key Agreement Protocol. Based on the above model and public keys management smart contract, we design a cross-domain authentication and key agreement protocol. The CAKA solves the cross-domain problem by querying the public key recorded on SCPKM, and it solves the authentication and key agreement problem by implementing DiffieHellman (DH) authenticated key exchange algorithm. The protocol is

Table 5: Update the public key in the smart contract.

Algorithm 3 update

function AP_Update (id, pk, t_begin, t_end, sig)
% CA node invokes the function to update public key.
if(sender.addr = = owner) % Guarantee only CA can invoke the function.
ap[id].PK = pk;
ap[id]. T_begin = *t*_begin;
ap[id]. T_end = *t*_end;
ap[id].sig = sig;
function user_update (hash1, hash2)
if(sender.addr = = owner)
Delete(user[hash1])
user[hash2] = true;

Table 6: Revoke the public key in the smart contract.

Algorithm 4 revoke
function AP_Revoke (id) % only CA can invoke the function to revoke public key of some node. if(sender.pk = = owner) Delete(ap[id]); function user_Revoke (hash) if(sender.pk = = owner) Delete(user[hash]);

shown in Figure 2 (denotes the blockchain). The specific process is as follows.

Suppose the user UE in domain U moves from home domain U to foreign domain V and it needs to authenticate an AP node VAP in the domain V and negotiate a session key to communicate securely with VAP (shown in Figure 2).

Step 1. UE sends a request message $\{T^U_{\mathrm{UE}}, \mathrm{DI}_U, \mathrm{ID}_{\mathrm{VAP}}, B^V, c_1\}$ to VAP

UE invokes get_CA( ) and get AP(VAP) of the smart contract in domain V. Then, the blockchain will return VAP$'s$ public key $\mathrm{PK}^V_{\mathrm{VAP}}$ and system parameters $\mathrm{basicpare}^V$ of domain V. Then UE generates an authentication public and private key pair (B^V, b^V) using Gen_V and picks a random nonce Nonce_1. UE computes the authentication key with VAP$k^V_{\mathrm{UE,VAP}} = H^V(b^V \cdot \mathrm{PK}^V_{\mathrm{VAP}})$ and encrypts Nonce_1 with the key $k^V_{\mathrm{UE,VAP}}$ to obtain $c_1 = E_{k^V_{\mathrm{UE,VAP}}}(T^U_{\mathrm{UE}}, \mathrm{DI}_U, \mathrm{ID}_{\mathrm{VAP}}, \mathrm{Nonce}_1)$. At last, it sends message $\{T^U_{\mathrm{UE}}, \mathrm{DI}_U, \mathrm{ID}_{\mathrm{VAP}}, B^V, c_1\}$ to VAP.

Step 2. VAP sends response message $\{B^U, c_2\}$ to UE.

Upon receiving message $\{T^U_{\mathrm{UE}}, \mathrm{DI}_U, \mathrm{ID}_{\mathrm{VAP}}, B^V, c_1\}$ from UE, VAP computes $h = H(H^U(\mathrm{PK}^U_{\mathrm{UE}}))$ ($\mathrm{PK}^U_{\mathrm{UE}}$ is in T^U_{UE}). Then, VAP invokes get_CA( ) and user_Verificate(h) of the smart contract in domain U. And, the blockchain will return system parameters $\mathrm{basicpare}^U$ of domain U and $d \in \{0, 1\}$ for user_Verificate(h). VAP verifies T^U_{UE} (verify the signature in T_U, $d = 1$ and $t_{\mathrm{begin}} \le t_{\mathrm{now}} \le t_{\mathrm{end}}$). If the verification is correct, VAP generates an authentication public and private key pair (B^U, b^U), picks a random nonce Nonce_2, and computes authentication keys with UE $k^V_{\mathrm{VAP,UE}} = H^V(s^V_{\mathrm{VAP}} \cdot B^V)$ and $k^U_{\mathrm{VAP,UE}} = H^U(b^U \cdot \mathrm{PK}^U_{\mathrm{UE}})$. It is obvious that $k^V_{\mathrm{VAP,UE}} = k^V_{\mathrm{UE,VAP}}$ holds by the following equations:

$$k^V_{\mathrm{UE,VAP}} = H^V\left(b^V \cdot \mathrm{PK}^V_{\mathrm{VAP}}\right) = H^V\left(b^V \cdot s^V_{\mathrm{VAP}} \cdot P^V\right) = H^V\left(s^V_{\mathrm{VAP}} \cdot B^V\right) = k^V_{\mathrm{VAP,UE}}, \tag{2}$$

VAP decrypts ciphertext c_1 with $k^V_{\mathrm{VAP,UE}}$ and obtains $T^{U'}_{\mathrm{UE}}, DI'_U, \mathrm{ID}_{\mathrm{VAP}}', \mathrm{Nonce}_1$. Then, VAP checks whether $T^{U'}_{\mathrm{UE}} = T^U_{\mathrm{UE}}, \mathrm{DI}'_U = \mathrm{DI}_U$, $\mathrm{ID}_{\mathrm{VAP}}' = \mathrm{ID}_{\mathrm{VAP}}$ hold. If all checks pass, VAP computes the session key $sk_{\mathrm{VAP,UE}} = H(k^U_{\mathrm{VAP,UE}} \| k^V_{\mathrm{VAP,UE}} \| \mathrm{Nonce}_1 \| \mathrm{Nonce}_2)$ and encrypts $\mathrm{Nonce}_1 \| \mathrm{Nonce}_2$ with the key $k^U_{\mathrm{VAP,UE}}$ to obtain $c_2 = E_{k^U_{\mathrm{VAP,UE}}}(\mathrm{Nonce}_1 \| \mathrm{Nonce}_2)$. Finally, VAP sends message $\{B^U, c_2\}$ to UE.

Step 3. UE sends session key confirmation message $\{\mathrm{ID}_{\mathrm{VAP}}, h, c_3\}$ to VAP.

Upon receiving message $\{B^U, c_2\}$ from VAP, UE computes another authentication key with VAP $k^U_{\mathrm{UE,VAP}} = H^U(s^U_{\mathrm{UE}} \cdot B^U)$. It is obvious that $k^U_{\mathrm{UE,VAP}} = k^U_{\mathrm{VAP,UE}}$ by the following equations:

$$k^U_{\mathrm{UE,VAP}} = H^U\left(s^U_{\mathrm{UE}} \cdot B^U\right) = H^U\left(s^U_{\mathrm{UE}} \cdot b^U \cdot P^U\right) = H^U\left(b^U \cdot B^U\right) = k^U_{\mathrm{VAP,UE}}. \tag{3}$$

Then, UE decrypts ciphertext c_2 with $k^U_{\mathrm{UE,VAP}}$ and obtains Nonce_1, Nonce_2, and T^V_{UE}. If Nonce_1 is the one which UE has sent to VAP, UE authenticates VAP. UE computes the session key $\mathrm{sk}_{\mathrm{UE,VAP}} = H(k^U_{\mathrm{UE,VAP}} \| k^V_{\mathrm{UE,VAP}} \| \mathrm{Nonce}_1 \| \mathrm{Nonce}_2)$ and encrypts $\mathrm{ID}_{\mathrm{VAP}} \| h$ with the key $\mathrm{sk}_{\mathrm{UE,VAP}}$ to obtain $c_3 = E_{\mathrm{sk}_{\mathrm{UE,VAP}}}(\mathrm{ID}_{\mathrm{VAP}} \| h)$. Finally, UE sends message $\{\mathrm{ID}_{\mathrm{VAP}}, h, c_3\}$ to UE. If Nonce_2 UE got is the one which VAP has sent, then UE and VAP both get the same session key sk:

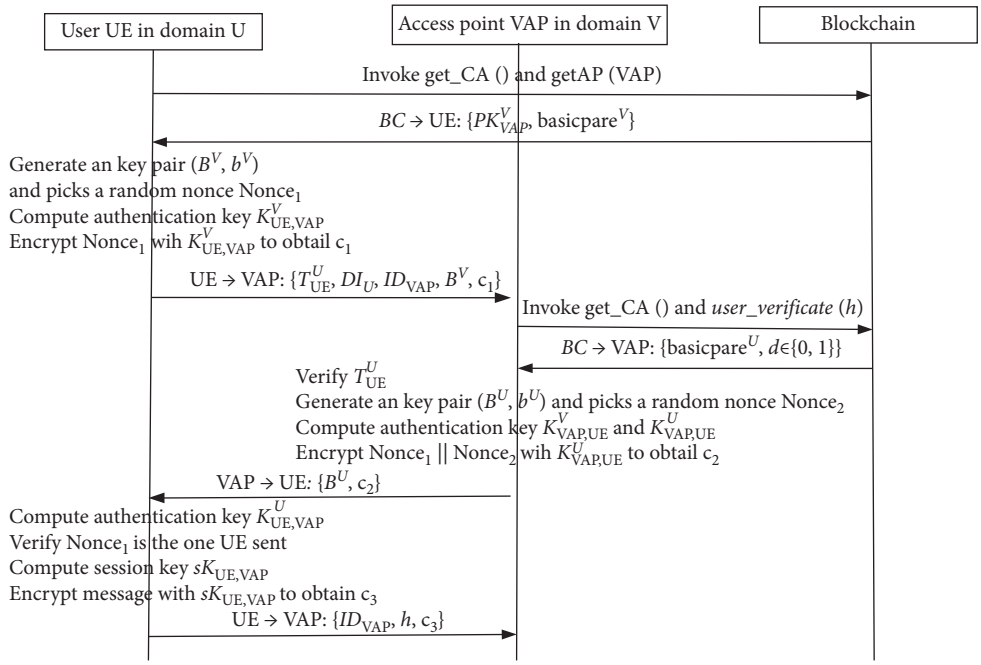

Figure 2: The workflow of cross-domain authentication and key agreement protocol.

$$\begin{aligned} \mathrm{sk} = \mathrm{sk}_{\mathrm{VAP,UE}} &= H\left(k^U_{\mathrm{VAP,UE}} \| k^V_{\mathrm{VAP,UE}} \| \mathrm{Nonce}_1 \| \mathrm{Nonce}_2\right) \\ &= H\left(k^U_{\mathrm{VAP,UE}} \| k^V_{\mathrm{VAP,UE}} \| \mathrm{Nonce}_1 \| \mathrm{Nonce}_2\right) \\ &= \mathrm{sk}_{\mathrm{UE,VAP}}. \end{aligned} \tag{4}$$

Step 4. After receiving message $\{ID_{VAP}, h, c_3\}$ from UE, VAP decrypts ciphertext c_3 using $\mathrm{sk}_{\mathrm{VAP,UE}}$. If decryption is successful, it confirms that UE got the right session key and authenticates UE.

UE and VAP belong to different domains using different parameters, but through the above CAKA protocol, they can authenticate each other and negotiate the session key to achieve secure communication.

4. Security Analysis

Security of our proposed cross-domain authentication and key agreement protocol are studied with the following respects.

4.1. Provable Security Analysis. This section proves the security of CAKA based on the CK security model. We first present a SK-secure protocol in AM. Then, we construct MT authenticators. Then, we apply the authenticators to the protocol in AM and get our protocol CAKA after necessary message reorganization and optimization. According to Theorem 1, our protocol is also SK-secure in UM.

4.1.1. Protocol π in AM. The specific process is as follows (Figure 3).

Step 1. The UE (T^U_{UE} is its authentication ticket) obtains the parameter pare^V of the domain V and the public key $\mathrm{PK}^V_{\mathrm{VAP}}$ of the VAP through querying contract, then it generates a public-private key pair (B^V, b^V), randomly picks a nonce Nonce_1, and sends the message $\{T^U_{\mathrm{UE}}, \mathrm{DI}_U, \mathrm{ID}_{\mathrm{VAP}}, B^V, \mathrm{Nonce}_1\}$ to the VAP.

Step 2. After receiving the message sent by the UE, the VAP firstly computes the hash $h = H(H^U(\mathrm{PK}^U_{\mathrm{UE}}))$ and then queries the contract to verify h and gets parameter pare^U of the domain U. After verifying the validity of T^U_{UE}, an authentication public and private key pair (B^U, b^U) is generated and a nonce Nonce_2 is randomly picked. Finally, the message $\{B^U, \mathrm{Nonce}_1, \mathrm{Nonce}_2\}$ is sent to the UE. The VAP can use B^V and $\mathrm{PK}^U_{\mathrm{UE}}$ to generate its session key $\mathrm{sk}_{\mathrm{VAS,UE}}$ with the UE as follows: $k^V_{\mathrm{VAP,UE}} = H^V(s^V_{\mathrm{VAP}} \cdot B^V)$, $k^U_{\mathrm{VAP,UE}} = H^U(b^U \cdot \mathrm{PK}^U_{\mathrm{UE}})$, and $\mathrm{sk}_{\mathrm{VAP,UE}} = H(k^U_{\mathrm{VAP,UE}} \| k^V_{\mathrm{VAP,UE}} \| \mathrm{Nonce}_1 \| \mathrm{Nonce}_2)$.

Step 3. After receiving the message $\{B^U, \mathrm{Nonce}_1, \mathrm{Nonce}_2\}$, the UE checks whether Nonce_1 is previously sent by itself, and if so, the UE completes the authentication with the VAP. The UE can use $\mathrm{PK}^V_{\mathrm{VAP}}$ and B^U to generate its session key $\mathrm{sk}_{\mathrm{UE,VAS}}$ with the VAP as follows: $k^V_{\mathrm{UE,VAP}} = H^V(b^V \cdot \mathrm{PK}^V_{\mathrm{VAP}})$, $k^U_{\mathrm{UE,VAP}} = H^U(s^U_{\mathrm{UE}} \cdot B^U)$, and $\mathrm{sk}_{\mathrm{UE,VAP}} = H(k^U_{\mathrm{UE,VAP}} \| k^V_{\mathrm{UE,VAP}} \| \mathrm{Nonce}_1 \| \mathrm{Nonce}_2)$. Then, UE uses $\mathrm{sk}_{\mathrm{UE,VAP}}$ to encrypt $\mathrm{ID}_{\mathrm{VAP}}$ and h and gets the ciphertext $c_3 = E_{\mathrm{sk}_{\mathrm{UE,VAP}}}(\mathrm{ID}_{\mathrm{VAP}} \| h)$. Then, it sends the message $\{\mathrm{ID}_{\mathrm{VAP}}, h, c_3\}$ to VAP.

So far, the UE and the VAP complete the authentication and key agreement.

Theorem 2. *If the CDH assumption is true and H is a random oracle, the protocol π is session key secure in AM.*

Proof. Firstly, it is easy to prove that the protocol π satisfies the first condition of session key secure in Definition 2. Namely,

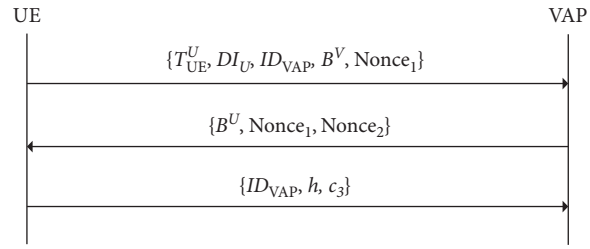

Figure 3: The workflow of protocol π in AM.

after the protocol is executed, the matching session computes the same session key by equations (2)–(4) and the session key is evenly distributed according to the hash property of H.

Next, it is proved that the protocol also satisfies the second condition of session key secure in Definition 2. In this paper, the algorithm $\mathcal{P}$ is constructed according to the idea of paper [30]. The algorithm $\mathcal{P}$ uses the adversary $\mathcal{A}$ as a subroutine to simulate the execution process of the protocol and answer all the queries and return the output message of the protocol to $\mathcal{A}$. The reduction to absurdity is used to prove that the protocol π satisfies condition 2 in the AM.

Suppose there is an adversary $\mathcal{A}$. Let ε be a nonnegligible advantage of distinguishing between a session key and a random number of the same length. The session key sk cannot be directly obtained, which can only be acquired by hashing obtained constituent elements. The sk is computed by $H(k^U_{UE,VAS} \| k^V_{UE,VAS} \| Nonce_1 \| Nonce_2)$, where PK^V_{VAP}, PK^U_{UE}, B^V, B^U, $Nonce_1$, and $Nonce_2$ are transmitted in clear text, which is easily obtained by $\mathcal{A}$, so the focus of the attack is $k^U_{UE,VAS}$ and $k^V_{UE,VAS}$, where $k^V_{UE,VAS} = H^V(s^V_{VAS} \cdot B^V) = H^V(b^V \cdot PK^V_{VAS})$ and $k^U_{UE,VAS} = H^U(s^U_{UE} \cdot B^U) = H^U(b^U \cdot PK^U_{UE})$. The advantage of solving the CDH problem for adversary $\mathcal{A}$ is denoted as ε_{CDH}. The probability of $\mathcal{P}$ guessing the test session is at least $1/L$ (L is number of sessions), and the probability of not guessing the test session is $1 - 1/L$. And, suppose in a test-session query, probability of guessing b is $1/2 + \varepsilon$. So, the probability of guessing b is $\Pr[b = b'] = (1/2 + \varepsilon) \times (1/L) + (1/2) \times (1/1 - L) = (1/2) + (\varepsilon/L)$. $\mathcal{P}$ can guess b by the following two cases: (i) completely randomly guess b; the probability is 1/2; (ii) solve the CDH problem. Then, $\Pr[b = b'] \le \varepsilon_{CDH} + 1/2$, so $\varepsilon/L = \Pr[b = b'] - 1/2 \le \varepsilon_{CDH}$. If ε is not negligible, ε_{CDH} is not negligible and obviously it contradicts the CDH assumption. So, the protocol π can be proved to meet the second condition.

Therefore, protocol π is session key secure in AM. □

4.1.2. Construct MT Authenticators. In this protocol, the UE authenticates the VAP and the VAP authenticates the UE, so two MT authenticators $\lambda_{SC,ENC}$ (encryption authenticator based on smart contract) and $\lambda'_{SC,ENC}$ are required.

MT authenticator $\lambda_{SC,ENC}$: the UE obtains the parameter $pare^V$ of the domain V and the public key PK^V_{VAP} of the VAP through inquiring the contract and computes the authentication key $k^V_{UE,VAP} = H^V(b^V \cdot PK^V_{VAP})$ using the newly generated private key b^V. Then, it randomly picks $Nonce_1$, computes ciphertext $c_1 = E_{k^V_{UE,VAP}}(T^U_{UE}, DI_U, ID_{VAP}, Nonce_1)$, and then sends the message $M_1 = \{T^U_{UE}, DI_U, ID_{VAP}, B^V, c_1\}$ to VAP.

After receiving the message, the VAP computes the authentication key $k^V_{VAP,UE} = H^V(s^V_{VAP} \cdot B^V)$ and $k^U_{VAP,UE} = H^U(b^U \cdot PK^U_{UE})$($PK^U_{UE}$ is in T^U_{UE}). Since $k^V_{VAP,UE} = k^V_{UE,VAP}$ (equation (2) in Section 3.4), the key $k^V_{VAP,UE}$ can be used to decrypt c_1 to obtain $Nonce_1$. VAP computes the ciphertext $c_2 = E_{k^U_{VAP,UE}}(Nonce_1 \| m)$, and sends the message $M_2 = \{B^U, c_2\}$ to UE.

After receiving the message, the UE computes the authentication key $k^U_{UE,VAP} = H^U(s^U_{UE} \cdot B^U)$ and then decrypts the ciphertext to obtain $Nonce_1$. Finally it checks whether $Nonce_1$ was previously sent to the VAP, and if so, the UE completes the authentication of VAP.

Here, only the MT authenticator $\lambda_{SC,ENC}$ is briefly described. As $\lambda'_{SC,ENC}$ is similar, we will not elaborate further.

Theorem 3. *If H^V and H are both random oracles, the CDH assumption is true, and the symmetric encryption algorithm E can resist selection message attacks; $\lambda_{SC,ENC}$ is the MT authenticator.*

Proof. Let $\mathcal{U}$ be the UM adversary who interacts with $\lambda_{SC,ENC}$. We construct an AM adversary $\mathcal{A}$ so that the outputs of $\mathcal{U}$ and $\mathcal{A}$ are the same except for the negligible probability. $\mathcal{A}$ initializes the protocol $\lambda_{SC,ENC}$ by selecting the key for a series of entities executing $\lambda_{SC,ENC}$ according to the running conditions of $\lambda_{SC,ENC}$. In the interaction with entities executing protocol $\lambda_{SC,ENC}$, adversary $\mathcal{A}$ runs $\mathcal{U}$ as a routine. Then, the interaction process runs as per the following rules:

(i) As long as the adversary $\mathcal{U}$ in the UM activates an entity B' to send a message m to the entity A', the adversary $\mathcal{A}$ in the AM activates B to send a message m to A

(ii) When the simulated entity A' outputs that A' receives a message m from B', the adversary $\mathcal{A}$ activates A to output a similar message

(iii) As long as the adversary $\mathcal{U}$ destroys an entity, $\mathcal{A}$ destroys the corresponding entity in the AM and sends the message to $\mathcal{U}$

(iv) Finally, $\mathcal{A}$ outputs the output of $\mathcal{U}$

If entity UE is not captured, but the message (UE, VAP, and M_1) is not in the undelivered message set, which means that the message is forged by the attacker. We call this event E. We want to prove that the probability of event E occurring is negligible. The adversary $\mathcal{U}$ forged a message that passed validation. This situation is true unless there are two kinds of events occurring: $\mathbb{E}_1$, the adversary $\mathcal{U}$ successfully falsifies the ciphertext without knowing the key $k^V_{UE,VAP}$, that is, $\mathcal{U}$ breaks the symmetric encryption algorithm; $\mathbb{E}_2$, the attacker computes $H^V(s^V_{VAP} \cdot B^V)$ or $H^V(b^V \cdot PK^V_{VAP})$ to get $k^V_{VAP,UE}$ without knowing private key of VAP or UE, that is, the CDH problem is solved. Since $E_1 \cup E_2 \supseteq E$, $\Pr(E_1) + \Pr(E_2) \ge \varepsilon$. If ε is not negligible, the probability that at least one of E_1 and E_2 occurs is not negligible. We can use $\mathcal{U}$ to construct an algorithm F to break through encryption algorithm or CDH problem with a probability of ε/C^2_n, where n is the number of entities that are activated. This contradicts the assumption

that the encryption algorithm E is safe and the CDH assumption is true. So, $\lambda_{SC,ENC}$ is the MT authenticator.

Similarly, $\lambda'_{SC,ENC}$ is the MT authenticator. So, the protocol CAKA is SK-secure in UM according to Theorem 1.

The provable security analysis shows that the protocol satisfies mutual authentication, key freshness, known key security, antireplay attack, and man-in-the-middle attack security. We also validated these security attributes in Section 3 using formal analysis tools. □

4.2. Analysis of Other Security Attributes

4.2.1. User Anonymity and Anonymity Controllability. Identity information of UE that is sent to VAP only include domain identifier and an authentication ticket containing the public key and the signature of CA. VAP is only sure of domain of UE and sure if the public key is registered but not identity by the information.

When the VAP is suspicious of the authentication information of the UE or after the UE roams into the foreign domain, the malicious anonymous access behavior may occur, VAP needs to submit the authentication ticket and related public information to the CA of UE′s domain for anonymous identity tracking. The CA first verifies the validity of the anonymous tracking information, and then, by querying the stored data to provide the identity information (ID_{UE}, PK_{UE}, h) (where h is in the authentication ticket) of UE. After receiving the response information of the CA, the VAP verifies that the equation $H(ID_{UE} \| H^U(PK_{UE})) = h$ is true. If so, the CA provides accurate information and VAP knows the true identity of UE. Otherwise, the information of UE provided is incorrect and VAP requires CA to continue to provide relevant information, that is, CA cannot protect malicious users.

4.2.2. Perfect Forward Secrecy. The random temporary numbers are unpredictable for any party except UE and VAP, because UE and VAP use new authentication private keys in every authentication process. Even if the adversary attacks secret information of UE and VAP or even captures the CA and obtains the long-term private key of UE and VAP, he cannot obtain the past temporary keys and the past encrypted random temporary numbers $Nonce_1$ and $Nonce_2$. And, he cannot get the past session keys certainly. Therefore, the scheme has the property of perfect forward secrecy.

4.2.3. Resilience to DDoS. The distributed architecture of blockchain naturally has point-to-point and multi-redundancy characteristics. Even if one node fails, other nodes are not affected, so there is no single-point failure problem. It is much more flexible than a centralized system in terms of denial of service attacks. Once a node fails, users connected to the failed node cannot enter the system.

In addition, based on the analysis of computing overhead for both parties of the protocol, as shown in Table 7, the difference between UE and VAP computing overhead is not significant. And, VAP checks for user identity and avoids replay attacks through receiving only messages with fresh $Nonce_1$. So, the protocol resists DDoS attacks.

4.3. Formal Analysis Tools. Compared with other mainstream protocol formal analysis tools, the formal analysis tool Scyther has the advantage that it can give explicit termination for the protocol with infinite session and infinite state set. The Scyther tools are based on the model detection algorithm and have a clear description of the state set trajectory. Scyther based on SPDL description language provides graphical attack output for both finite and infinite sessions. Scyther series tools include Scyther and Scyther-Compromise, among which Scyther-Compromise tools use a variety of adversary enquiry capabilities under the strong security model as tick options, including forward security, weak forward security, perfect forward security, temporary key leakage, state leakage, and other strong security attributes. In the button-based human-computer interaction interface, as long as different combinations of different queries are selected, the protocol is analyzed under different strong security models such as CK or eCK. However, the Scyther tools do not include embedded algebraic operation properties and cannot formally describe algebraic properties, making it difficult to find attacks that involve complex algebraic operations.

AVISPA (automated validation of internet security-sensitive protocols and applications) is an automated validation tool of network security protocol that uses HLPSL formal language to describe target protocols. HLPSL is a modular, role-based formal language that describes the specified control flow patterns and data structures. HLPSL describes attacker models and complex security attributes in AVISPA. The HLPSL2IF algorithm converts the protocol file into a .cpp type file written by an if statement and then calls the four backend analysis tools OFMC, CL-AtSe, SATMC, and TA4SP to verify the security of the protocol. Through these four background analysis operators, AVISPA tools have excellent scope and scalability and can analyze large-scale network security protocols and establish a complex formal model for protocol processes, security targets, and attack trajectories. At the same time, it has high computational efficiency. However, the types of models covered by AVISPA tools are limited, and the supported security models are relatively simple. It is difficult to give complete results for the analysis of security protocols under strong security models.

Considering the advantages and disadvantages of Scyther and AVISPA tools, to give a comprehensive and objective formal analysis of the proposed protocol CAKA, this paper uses the combination of Scyther tool and AVISPA tool to analyze the security of the protocol. This can avoid the attack omission which is caused by the algebraic operation property defect of Scyther tool and the security model defect of AVISPA tool. The proposed protocol CAKA is formally described by HLPSL language in Figure 4, in which the role user node is defined as the left algorithm (sigama_Init) of Figure 4, the role AP is defined as the right picture of Figure 4 (sigma_Resp), and the key security and authentication of the protocol are analyzed under the Dolev-Yao security model. The analysis results are shown in Figure 5. The results show that AVISPA's OFMC

Table 7: Computing overhead for UE and VAP of the protocol.

	UE	VAP
Verification	0	1
Symmetric encryption/decryption	1/2	2/1
DH	2	2
Key pair generation	1	1

```
role sigma_Resp (B,A : agent,
                 G: text,
                 H: hash_func,
                 Ka,Kb : public_key,
                 Snd,Rcv : channel(dy))
played_by B
def=% Message authentication (G2)
    local State : nat,
        Y,N2,N1 : text,
        GX : message,
        MK_A,MK_B : message

 const one : text,
 sec_r_MK_B : protocol_id
 nit State := 0

 transition
  1. State = 0  /\ Rcv(GX'.{N1'.GX'}_H(exp(GX',inv(Kb)))) =|>
    State':= 1 /\ Y' := new()
              /\ N2' := new()
              /\ MK_B' :=H(N1.N2'.exp(Ka,Y').exp(GX,Y'))
              /\ MK_A' :=MK_B'
              /\ Snd(exp(G,Y').{N2'.exp(G,Y')}_H(exp(Ka,Y')))
              /\ secret(MK_B',sec_r_MK_B,{A,B})
              /\ witness(B,A,mk_a,MK_B')
              /\ request(B,A,mk_b,MK_A')

end role
```

```
role sigma_Init (A,B : agent,
                 G : text,
                 H : hash_func,
                 Ka,Kb : public_key,
                 Snd,Rcv : channel(dy))
played_by A
def=
   local State : nat,
      X,N1,N2 : text,
      GY : message,
      MK_A,MK_B : message

   const two: text,
      sec_i_MK_A: protocol_id

   init State := 0

   transition
  1. State = 0  /\ Rcv(start) =|>
    State':= 1 /\ X' := new()
              /\ N1' := new()
              /\ Snd(exp(G,X').{N1'.exp(G,X')}_(H(exp(Kb,X'))))

  2. State = 1  /\ Rcv(GY'.{N2'.GY'}_H(exp(GY',inv(Ka)))) =|>
    State':= 2 /\ MK_A' :=H(N1.N2'.exp(Kb,X).exp(GY',X))
              /\ MK_B' :=MK_A'
                        /\ secret(MK_A',sec_i_MK_A,{A,B})
                        /\ request(A,B,mk_a,MK_B')
                        /\ witness(A,B,mk_b,MK_A')

end role
```

Figure 4: HLPSL role definition in CAKA protocol.

engine displays the analysis results as "safe" in 0.02 seconds, and CL-AtSe engine displays the analysis results as "safe" in negligible seconds. As the backend SATMC and TA4SP do not contain the algebraic properties of exponent operation such that these two backends cannot handle the analysis of this scheme. As shown in Figure 6, the SPLA language is used to describe the CAKA protocol, and we use the Scyther-Compromise tool to analyze it under the CK security model by checking the options "Long-Term Key Reveal," "wPFS," "Session-Key Reveal," and "State Reveal" in the analysis options as shown in Figure 7. The Scyther-Compromise tool shows that the protocol is session key secure under the CK model, as shown in Figure 8.

5. Performance Evaluation

Since the nodes register in the form of a blockchain transaction (invoking the smart contract), we do not consider computation overhead and transaction fees of blockchains, and we only briefly analyze the performance of authentication and key agreement protocol.

We analyzed the performance of several typical cross-domain authentication schemes Jeon et al. [37], Huo et al. [38] and ours by analyzing message transmission times and computation cost. Table 8 compares the message transmission times between the nodes (HA is home AP and TA is target AP) in three protocols. As can be seen from the table, our solution has obvious advantages. Our scheme can accomplish two-way authentication only by transmitting messages three times between users and target AP, without the assistance of home server or AP. The other two schemes need to forward messages through home AP, which increases the communication delay, resulting in a total of four messages to be transmitted. So, our scheme has less communication delay.

For computational overhead, we use the OpenSSL library to program calculations using the C program language. Our experimental environment is Ubuntu 18.04 with Intel (*R*) Core (TM) i7-6700 CPU @ 3.40 GHZ CPU and 4 GB RAM memory. We measure the approximate time cost of cryptography operations through the OpenSSL library, where ECDH, ECDSA, ECIES, and elliptic curve key pair generation are measured on the curve ANSI X9.62 prime192v1. The results are presented in Table 9. It can be seen from Table 9 that public key cryptography (signature, encryption, and key pair generation) takes more time, while symmetric cryptography and hashing take less time. The difference between them is more than 40 times. The times of cryptographic operations of the three protocols are compared in Table 10. As can be seen from the table, our solution requires less public key encryption and less computational latency. Combining Tables 9 and 10, we calculate calculation

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/caka.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.02s
  visitedNodes: 4 nodes
  depth: 2 plies
```

```
SUMMARY
SAFE

DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL

PROTOCOL
/home/usr/local/avispa-1.1/testsuite/results/text.if

GOAL
As Specified

BACKEND
CL-AtSe

STATISTICS

Analysed  : 3 states
Reachable  : 0 states
Translation: 0.00 seconds
Computation: 0.00 seconds
```

Figure 5: The analysis results of AVISPA.

```
// The protocol description

symmetric-role protocol CAKA-A(I,R)
{
        role I
        {
                fresh x,n1: Nonce;
                var Y,K,n2: Ticket;

                send_1(I,R, I, n1,g1(x), {I, n1,g1(x)}sk(I));
                recv_2(R,I,R, n1,n2, Y, {R, n1,n2,Y}sk(R));

                claim(I,SKR,    KDF(exp(g1(sk(R)),x),exp(Y,sk(I)),n1,n2));

        }

        role R
        {
                fresh y,n2: Nonce;
                var X,K,n1: Ticket;

                recv_1(I,R, I, n1,X, {I, n1,X}sk(I) );
                send_2(R,I, R, n1,n2,g1(y), {R, n1,n2,g1(y)}sk(R));

                claim(R,SKR,    KDF(exp(X,sk(R)), exp(g1(sk(I)),y),n1,n2) );
        }
}
```

Figure 6: The SPDL protocol description of CAKA protocol.

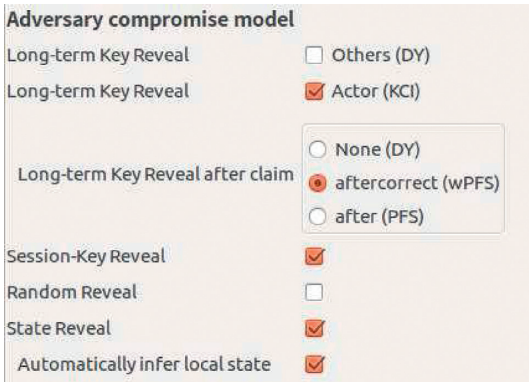

Figure 7: The setting of Scyther-Compromise.

Figure 8: The analysis results of Scyther-Compromise.

overheads of the three protocols as 2.31 ms, 2.22 ms, and 0.355 ms, respectively. Clearly, the calculation cost of our solution is very low, less than 1/6 of the other two schemes. So, our scheme has less communication delay. In addition, the computation cost of UE and TA computed from Tables 7 and 10 is 0.1338 ms and 0.2213 ms, respectively.

Our solution communication and computation overhead are relatively small, especially the computation overhead and the performance advantages are obvious. So, our solution not only achieves secure cross-domain authentication but also enables fast real-time authentication.

Table 8: Communication overhead (transmission times).

Protocol	UE and TA	TA and HA	Total
Jeon	2	2	4
Huo	2	2	4
Ours	3	0	3

Table 9: Computation cost of cryptographic operation.

Cryptographic operation	Time cost (ms)
Signature (ECDSA)	0.04418
Verification (ECDSA)	0.08782
Symmetric encryption/decryption (AES)	0.001207/0.001528
Asymmetric encryption/decryption (ECIES)	1.05103/0.843882
Hash (SHA256)	0.000900
ECDH	0.05408
Key pair generation	0.017789

Table 10: Comparison the times of cryptographic operation.

Protocol	Signature	Verification	Asymmetric encryption/decryption	DH	Key pair generation	Symmetric encryption/ decryption	Hash	Total costs (ms)
Jeon	3	3	1/1	0	0	2/2	13	2.31
Huo	2	2	1/1	0	3	1/1	5	2.22
Ours	0	1	0	4	2	3/3	8	0.355

6. Conclusion

This paper proposes a cross-domain authentication and key agreement system based on smart contract for heterogeneous wireless networks. In the solution, all security domains join into the consortium chain, and the CA in each domain manages the public key through the smart contracts. We implement mutual cross-domain authentication and provide user anonymity in the solution. The protocol CAKA is proved secure under the CK model and two formal analysis tools Scyther tool and AVISPA also report the protocol is safe. Without public key encryption and signature, the protocol improves the efficiency of cross-domain authentication compared with some existed ones. Moreover, the system is based on the design of the consortium chain and it has strong scalability.

The system designed in this paper only tests its computational consumption and does not perform simulation experiments on the whole system to test other performance such as communication. In the future, it is necessary to study the use of network simulation software OPNET or the actual wireless network system for more detailed system evaluation.

Data Availability

This paper uses the combination of Scyther tool and AVISPA tool to analyze the security of the protocol. The approximate time cost of cryptography operations is measured through the OpenSSL library. The Scyther tool can be downloaded from the website https://people.cispa.io/cas.cremers/scyther/. The AVISPA tool can be downloaded from the website http://www.avispa-project.org/. The OpenSSL library can be downloaded from the website https://www.openssl.org/.

Conflicts of Interest

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

Acknowledgments

This work was supported by the Innovative Research Groups of the National Natural Science Foundation of China (61521003) and Youth Program of National Natural Science Foundation of China (61502533).

References

[1] X. Duan and X. Wang, "Authentication handover and privacy protection in 5G HetNets using software-defined networking," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 28–35, 2015.

[2] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion," *Ad Hoc Networks*, vol. 20, no. 2, pp. 96–112, 2014.

[3] M. K. Mahshid and R. Eslamipoor, "An efficient and secure authentication for inter-roaming in wireless heterogeneous network," *Social Network Analysis & Mining*, vol. 4, no. 1, pp. 222–114, 2014.

[4] G. L. Millán, M. G. Pérez, G. M. Pérez, and A. F. G. Skarmeta, "PKI-based trust management in inter-domain scenarios," *Computers & Security*, vol. 29, no. 2, pp. 278–290, 2010.

[5] H.-X. Peng, "An identity-based authentication model for multi-domain," *Chinese Journal of Computers*, vol. 8, no. 29, pp. 1271–1281, 2006.

[6] R. C.-W. Phan, "Non-repudiable authentication and billing architecture for wireless mesh networks," *Wireless Networks*, vol. 17, no. 4, pp. 1055–1061, 2011.

[7] T. Gao, G. Nan, and K. Yim, "LEAS: localized efficient authentication scheme for multi-operator wireless mesh network with identity-based proxy signature," *Mathematical & Computer Modelling*, vol. 58, no. 5-6, pp. 1427–1440, 2013.

[8] L. Chen, Z. Cheng, and N. P. Smart, "Identity-based key agreement protocols from pairings," *International Journal of Information Security*, vol. 6, no. 4, pp. 213–241, 2007.

[9] N. McCullagh and P. S. Barreto, "A new two-party identity-based authenticated key agreement," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 262–274, Springer, San Francisco, CA, USA, February 2005.

[10] R. Yasmin, E. Ritter, and G. Wang, "A pairing-free ID-based one-pass authenticated key establishment protocol for wireless sensor networks," in *Proceedings of the Fifth International Conference on Sensor Technologies and Applications (SENSORCOMM'11)*, Nice/Saint Laurent du Va, France, August 2011.

[11] M. C. Gorantla, C. Boyd, and J. M. González Nieto, "ID-based one-pass authenticated key establishment," vol. 81, pp. 39–46,in *Proceedings of the sixth Australasian conference on Information security, Australian Computer Society*, Inc., Brisbane, Australia, January 2008.

[12] C. Swanson and D. Jao, "A study of two-party certificateless authenticated key-agreement protocols," in *Proceedings of the International Conference on Cryptology in India*, pp. 57–71, Springer, New Delhi, India, December 2009.

[13] S. M. Cheng, C. H. Ho, S. Chen, and S. H. Chang, "Distributed anonymous authentication in heterogeneous networks," in *Proceedings of the Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 505–510, IEEE, Nicosia, Cyprus, August 2014.

[14] A. Fu, N. Qin, Y. Wang, Q. Li, and G. Zhang, "Nframe: a privacy-preserving with non-frameability handover authentication protocol based on (t, n) secret sharing for LTE/LTE-A networks," *Wireless Networks*, vol. 23, no. 7, pp. 2165–2176, 2017.

[15] Y. Yang, X. Zheng, X. Liu, S. Zhong, and V. Chang, "Cross-domain dynamic anonymous authenticated group key management with symptom-matching for e-health social system," *Future Generation Computer Systems*, vol. 84, pp. 160–176, 2018.

[16] Y. Lu, G. Xu, L. Li, and Y. Yang, "Anonymous three-factor authenticated key agreement for wireless sensor networks," *Wireless Networks*, vol. 25, no. 4, pp. 1461–1475, 2019.

[17] Q. Cheng, C. Hsu, and L. Harn, "Lightweight noninteractive membership authentication and group key establishment for WSNs," *Mathematical Problems in Engineering*, vol. 2020, Article ID 1452546, 9 pages, 2020.

[18] O. Arezou, A. Hamed, N. Morteza, and A. Dariush, "Three party secure data transmission in IoT networks through design of a lightweight authenticated key agreement scheme," *Future Generation Computer Systems*, vol. 100, pp. 882–892, 2019.

[19] M. Qi and J. Chen, "A privacy-preserving biometrics based authenticated key agreement scheme using ECC," *International Journal of Communication Systems*, vol. 33, no. 11, Article ID e4407, 2020.

[20] M. Akram, Z. Ghaffar, K. Mahmood, S. Kumari, K. Agarwal, and C. Chen, "An anonymous authenticated key-agreement scheme for multi-server infrastructure," *Human-Centric Computing and Information Sciences*, vol. 10, no. 1, 2020.

[21] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2011, http://bitcoin.org/bitcoin.pdf.

[22] C. Fromknecht, D. Velicanu, and S. Yakoubov, "Certcoin: a namecoin based decentralized authentication system," Technical Reports, 6, Massachusetts Institute of Technology, Cambridge, MA, USA, http://courses.csail.mit.edu/6.857/2014/files/19-fromknecht-velicann-yakoubov-certcoin.pdf, 2014.

[23] C. Fromknecht, D. Velicanu, and S. Yakoubov, *A Decentralized Public Key Infrastructure with Identity Retention*, IACR Cryptology ePrint Archive, Lyon, France, 2014, https://eprint.iacr.org/2014/803.pdf.

[24] L. Axon, "Privacy-awareness in blockchain-based PKI," in *CDT Technical Paper Series*, University of Oxford, Oxford, UK, 2015, https://ora.ox.ac.uk/objects/uuid:f8377b69-599b-4cae-8df0-f0cded53e63b.

[25] K. Lewison and F. Corella, *Backing Rich Credentials with a Blockchain PKI*, Technical Report, Pomcor, San Diego, CA, USA, https://pomcor.com/techreports/BlockchainPKI.pdf, 2016.

[26] W. Wang, N. Hu, and X. Liu, "BlockCAM: a blockchain-based cross-domain authentication model," in *Proceedings of the IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 896–901, IEEE, Guangzhou, China, June 2018.

[27] M. T. Hammi, P. Bellot, and A. Serhrouchni, "BCTrust: a decentralized authentication blockchain-based mechanism," in *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, Abu Dhabi, UAE, April 2018.

[28] M. Vukolić, "Rethinking permissioned blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pp. 3–7, ACM, Abu Dhabi, UAE, April 2017.

[29] G. E. Wood, "A secure decentralized generalised transaction ledger," 2014, http://gavwood.com/paper.pdf.

[30] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 453–474, Springer, Innsbruck, Austria, May 2001.

[31] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pp. 419–428, ACM, Dallas, TX, USA, May 1998.

[32] Y. S. T. Tin, C. Boyd, and J. G. Nieto, "Provably secure key exchange: an engineering approach," vol. 21, pp. 97–104, in *Proceedings of the Australasian Information Security WORKSHOP conference on ACSW Frontiers 2003*, Australian Computer Society, Inc., Adelaide, Australia, January 2003.

[33] D. Abbasinezhad-Mood, A. Ostad-Sharif, S. M. Mazinani, and M. Nikooghadam, "Provably-secure escrow-less Chebyshev chaotic map-based key agreement protocol for vehicle to grid connections with privacy protection," *IEEE Transactions on Industrial Informatics*, 2020.

[34] D. Abbasinezhad-Mood and M. Nikooghadam, "An anonymous ECC-based self-certified key distribution scheme for the smart grid," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7996–8004, 2018.

[35] D. Abbasinezhad-Mood and M. Nikooghadam, "Efficient anonymous password- authenticated key exchange protocol to read isolated smart meters by utilization of extended Chebyshev chaotic maps," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4815–4828, 2018.

[36] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos, "BSeIn: a blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, 2018.

[37] W. Jeon, Y. Lee, and D. Won, "A secure user-friendly authentication scheme with anonymity for wireless communications," in *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, pp. 1–7, ACM, Carrillo, Costa Rica, December 2013.

[38] S. W. Huo, C. Y. Luo, and H. Z. Xin, "Identity-based interdomain authentication scheme in pervasive computing environments," in *Proceedings of the Intelligent Computing and Information Science*, pp. 314–320, Springer, Chongqing, China, January 2011.