IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 December 2023, accepted 21 December 2023, date of publication 28 December 2023,
date of current version 9 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3347808

RESEARCH ARTICLE

A New Identity Authentication and Key Agreement Protocol Based on Multi-Layer Blockchain in Edge Computing

YAO CHEN, QINGQING YANG, XIN ZENG, DENGQI YANG, AND XIAOWEI LI
School of Mathematics and Computer Science, Dali University, Dali 671003, China

Corresponding author: Xiaowei Li (lixiaowei_xidian@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 62262001, Grant 61902049, and Grant 32260131.

ABSTRACT In today's interconnected world, identity authentication and key agreement are important links in the secure communication process of IoT terminal devices. In the edge computing environment, with the frequent cross-domain authentication and data sharing of IoT devices in different security domains, identity authentication faces a series of challenges and security issues. Most of the traditional identity authentication methods are based on public key infrastructure, which is prone to single point of failure and is not applicable to the distributed architecture of edge computing. In this article, we apply blockchain technology to the identity authentication and key agreement process of IoT terminal devices. In order to meet cross-domain requests from terminal devices in different security domains, a multi-layer blockchain authentication architecture is designed. The hash value of the digital certificate is stored on the blockchain and combined with dynamic accumulator technology to enhance the reliability and authentication efficiency of the digital certificate. Security analysis and experimental results demonstrate that our scheme can achieve efficient and secure authentication and key agreement.

INDEX TERMS Edge computing, identity authentication, key agreement, multi-layer blockchain.

I. INTRODUCTION

Nowadays, we are witnessing the rapid proliferation of intelligent devices in the digital age, ushering in an era of interconnected everything [1]. The IoT exhibits characteristics such as multi-source heterogeneity and openness, but it also confronts challenges related to cyber threats and privacy [2]. This makes the protection of the identity information of IoT devices of great significance to individuals, families, society and even national security. The applications of the Internet of Things span various domains, including the smart grid [3], smart cities [4], intelligent transportation [5], healthcare [6], and the industrial Internet of Things [7]. With the advancement of the Internet, a large number of terminal devices are being connected to the network. Traditional cloud centers suffer from data processing delays, resulting in a heavy burden on cloud center authentication. The traditional cloud computing environment falls short of meeting the growing demand. The introduction of edge computing [8] aims to alleviate the computing pressure on the cloud center by diverting data flow. Edge servers are deployed to take over certain functions of the cloud center and handle the computing and storage tasks of devices in close proximity to the terminals. The network architecture is illustrated in Fig. 1.

At present, traditional identity authentication methods are mostly based on Public Key Infrastructure (PKI) implementation, which belongs to centralized authentication [9], [10], [11]. The centralized authentication process requires the involvement of a trusted third party and is prone to single point of failure issues. The security of this type of authentication relies on the stability of the Certificate Authority (CA) [12]. The authentication process uses digital certificates issued by CAs to authenticate identities. Once a CA is attacked, it will result in identity authentication being unable

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang.

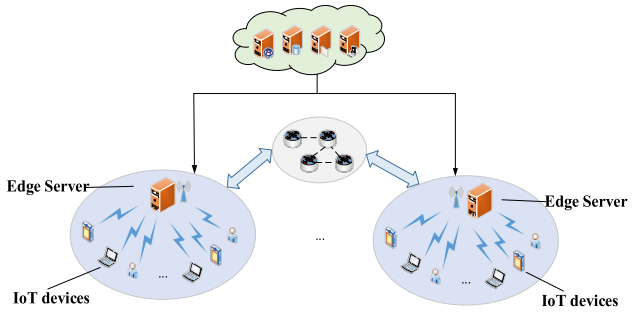


FIGURE 1. Edge computing architecture.

to proceed. Digital certificate verification in the authentication process needs to be completed by public key encryption and digital signature. This leads to complication of certificate verification process, longer verification times and verification time. There are multiple security domains for identity authentication in the edge computing environment [13], and the authentication hierarchy is complex. Massive terminal devices need to cross-domain authentication under different security domains to request frequent data access. In order to ensure the consistency and security of the distributed Internet of Things system, we provide a distributed authentication scheme using blockchain technology. Blockchain itself is a distributed network architecture. The binding energy of blockchain and edge computing can well solve the security problems in identity authentication [14], [15]. Due to its decentralized, tamper proof, open and transparent characteristics, blockchain technology can solve the single point of failure problem of public key infrastructure-based authentication [16]. Therefore, the use of blockchain technology can establish communication channels for IoT devices in different security domains.

The main contributions of this paper can be summarized as follows:

- This paper is primarily based on the distributed architecture of edge computing networks. In order to address the cross-domain requests from IoT devices in different security domains, we have designed a multi-layer blockchain authentication architecture and proposed a protocol scheme for identity authentication and key agreement for both single-domain and cross-domain terminal devices based on the multi-layer blockchain.
- The hash value of the digital certificate is stored on the blockchain, which improves the reliability of the digital certificate. It also simplifies the digital certificate reliability verification process, reduces the number of signature verification and improves authentication efficiency. To solve the problem of inefficient on-chain data queries due to the increased size of the blockchain, the authentication process incorporates dynamic accumulator technology to improve the efficiency of the authentication process certificate verification.
- The protocol designed in this paper was analyzed for security and performance. The results indicate that the protocol meets security requirements, as demonstrated by formal security analysis tools and proof of protocol security under the ROR model. Comparing its performance with similar cross-domain authentication protocols shows that this protocol exhibits good computational performance.

The rest of this article is organized as follows. In Section II, we investigated the work related to cross-domain authentication. Section III introduces relevant knowledge and system models. In Section IV, we propose our authentication and key agreement scheme based on the edge computing environment. The security analysis and Blockchain impleme-ntation are presented in Sections V and VI respectively. Section VII is the performance analysis of the protocol. Finally, we conclude the paper in Section VIII.

II. RELATED WORK

Traditional cloud centers interact with IoT devices directly for authentication and information. In a scenario where a large number of IoT devices are dynamically accessed, this will lead to high operating costs and difficulties in mutual trust between edge servers of different security domains. IoT application scenarios exhibit greater dynamism, heterogeneity, and scale compared to cloud computing environments. However, under the network architecture of edge computing, frequent cross-domain authentication and data sharing among terminal devices pose additional requirements for identity authentication and key agreement protocols. To address various security and performance issues, many scholars have conducted research in this area.

Garba et al. [17] proposed a blockchain based BB-PKI to manage certificates. To avoid single point of failure, multiple CAs issue certificates and record certificate transactions on the blockchain through smart contracts. In the work of Garba et al. [18], a blockchain-based cross-domain authentication scheme was proposed. It has privacy-preserving features for low-performance devices. A set of trusted CAs associated with a specific region in the blockchain is established and these CAs are responsible for issuing and renewing certificates. Gu and Chen [19] proposed a cross-domain authentication protocol based on blockchain, which stores certificate hash values on the blockchain and designs a certificate revocation process. However, there is a problem of low authentication efficiency. Wang et al. [20] proposed an anonymous authentication model in combination with blockchain technology. A decentralized network with the root certificate authority as the authentication node was constructed. Yuan et al. [21] proposed a cross-domain identity authentication scheme between the PKI domain and the IBE domain, in which a heterogeneous cross-domain authentication key agreement protocol was designed to achieve cross enterprise communication, but there was a problem of high computational complexity. In [22] and [23], an identity authentication protocol with privacy protection was proposed, which introduces group signature verification for users. However, this

TABLE 1. Main work and limitations.

Scheme	Main work	limitation
Gu *et al.* [19]	Blockchain-based cross-domain authentication protocol is designed.	inefficient certification
Yuan *et al.* [21]	Cross-domain authentication scheme between PKI domain and IBE domain is proposed.	high computational complexity
Scheme [22][23]	Authentication protocols with privacy preserving features are proposed and group signature verification is introd-uced.	high computational overhead
Moni *et al.* [29]	A lightweight WANET authentication protocol based on cuckoo filters is proposed.	probability of false alarm

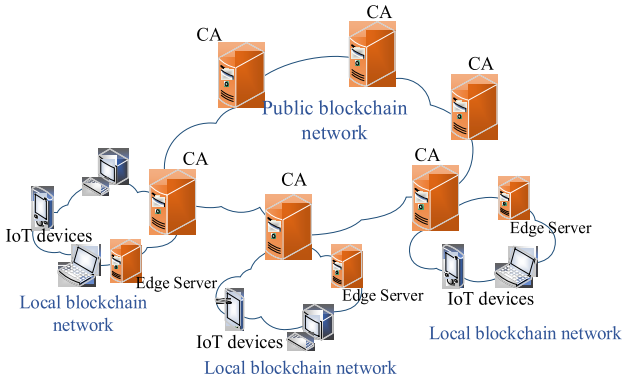


FIGURE 2. Multi-layer blockchain authentication architecture.

scheme has high computational overhead and is not suitable for low-performing IoT devices. Wang et al. [24] proposed a blockchain based on cross-domain authentication scheme for the Internet of Things. The authentication relationship is abstracted into an undirected graph, and the combination of accumulator and digital signature formulates the authentication problem, which can be well verified the legitimacy of the authentication. Wang et al. [25] established a multi-CA authentication architecture to achieve cross-domain certificate information sharing. The cross-domain certificate revocation mechanism was designed to improve the authentication efficiency. Jia et al. [26] proposed a decentralized authentication model using identity-based own authentication algorithm instead of PKI. The model is based on blockchain combined with smart contracts and threshold ciphers and has good flexibility. Guo et al. [27] focused on the authentication between different security domains in IoT. A master-slave blockchain architecture supporting distributed cross-domain authentication was designed. An improved Byzantine fault tolerance (RIBFT) based on reputation value model was proposed for trusted authentication and data traceability. The scheme considers the trustworthiness of the authentication process and the traceability of transaction information. Cheng et al. [28] proposed a two-way authentication scheme based on blockchain for edge servers and IoT devices. The scheme is mobile and anonymous, and it is suitable for use in collaborative edge computing environments. Moni and Manivannan [29] proposed a lightweight identity authentication protocol for WANET based on the cuckoo filter. Performance evaluation shows that this protocol has good performance overhead and faster authentication efficiency compared to other protocols, but there is a probability of false positives, making it difficult to adapt to cross-domain authentication in important situations. A summary of the work of the above scheme and its limitations are shown in Table 1.

However, most Internet of Things authentication protocols now have two shortcomings. First, authentication schemes based on traditional cloud centers are centralized authentication, which essentially relies on trusted third parties. This mode will cause single point of failure and scalability problems. Secondly, most existing protocols are not suitable for low-performing IoT devices, as IoT devices have limited resources and cannot bear significant computational costs. Therefore, we propose a new cross-domain identity authentication and key agreement protocol based on multi-layer blockchain, which combines dynamic accumulator to improve identity authentication efficiency and achieve efficient and secure authentication and key agreement.

III. SYSTEM ARCHITECTURE AND DYNAMIC ACCUMUL-ATOR TECHNOLOGY

A. SYSTEM ARCHITECTURE

Due to the existence of different security domains in the edge computing environment. The same domain terminal device authentication approach is no longer applicable to the authentication environment of multiple security domains. The local area blockchain in a single security domain cannot meet the demand of terminal devices to access resources across domains. It also cannot guarantee that the endpoints in other security domains can securely access the security domain. In order to ensure the consistency and security of decentralized IoT systems, we have designed a multi-layer blockchain architecture. In addition, multi-layer blockchains can effectively alleviate the storage pressure of individual blockchains, reduce the burden of blockchain queries, and meet the cross-domain authentication and key agreement between terminal devices in different security domains.

The system architecture of multi-layer blockchain designed in this paper is composed of local blockchain network and public blockchain network. Terminal devices, edge server, and local certificate authority CA belonging to the same security domain together form the local blockchain network. Certificate authorities in different security domains form a public blockchain network. When terminal devices belonging to different security domains need to authenticate communication, they can use the public blockchain network as a communication bridge. The local blockchain and public blockchain are built based on the prototype of the alliance chain, and only approved nodes are allowed to join the blockchain network. The system architecture is shown in Fig. 2.

Local blockchain is responsible for data security of nodes within the same security domain and identity authentication of the same security domain. Due to the limited computing performance of IoT devices, identity authentication and queries are allowed, but they do not participate in the maintenance of blockchain ledgers. The local blockchain stores the certificate hash value of terminal devices within the local domain. Compared to the public blockchain, there are fewer users and fewer certificates within the local blockchain, so lookups are more efficient. When cross domain authentication occurs, local devices cannot access blockchains from other domains. For this reason, we introduce public blockchain. The hashes of digital certificates in all secure domains are stored in the public blockchain, which can satisfy the verification of digital certificates during cross-domain authentication.

Unlike CA in traditional PKI, each security domain has a CA. CA, the certificate issuing authority, is both a node in the local blockchain and a member node of the public blockchain. CA performs identity verification and certificate issuance for IoT devices joining the blockchain in the local blockchain network, and it issues cross-domain certificates for cross-domain authenticated IoT devices in the public blockchain. CA calculates the certificate hash and cumulative value for each certificate issued. The dynamic accumulator verifies the hash of the packaged certificate on the chain, ensuring the safe and efficient identity authentication process.

The edge server not only participates in the maintenance of the local blockchain network, but also assists in completing cross-domain authentication of terminal devices in different security domains. The edge server within each security domain is involved in maintaining the local blockchain. At the time of cross-domain access request from terminal devices, the edge server and CA are jointly responsible for the information interaction for cross-domain authentication.

Algorithm 1 Create Accumulator

Input: security parameter 1^{λ}
Output: accumulator administrator keys (sk_{acc}, pk_{acc})
Choose Large prime number p, q
Compute $N = p * q$
Compute Euler function $\varphi(n) = (p-1) * (q-1)$
Select integer g
g satisfies $g^{\varphi(n)} \bmod n = 1$ and $g \neq 1$
$acc_0 = g \in Z_N$
$(sk_{acc}, pk_{acc}) = ((p, q), N) \leftarrow 1^{\lambda}$
Return (sk_{acc}, pk_{acc})

Algorithm 2 Add Element

Input: cumulative value acc_x, add element x,element collection X, pk_{acc}
Output: $acc_{x'}, X'$
if $x \notin X$ **then**
  $acc_{x'} = acc_{X \cup \{x\}} = acc_X^x \bmod N$
  $X' = X \cup \{x\}$
end
return $acc_{x'}, X'$

Algorithm 3 Del Element

Input: cumulative value acc_x, delete element x, element collection X, (sk_{acc}, pk_{acc})
Output: $acc_{x'}, X'$
if *Verelement*() **then**
  $acc_{x'} = acc_{X \setminus \{x\}} = acc_X^{x^{-1} \bmod \varphi(n)} \bmod N$
  $X' = X \setminus \{x\}$
end
return $acc_{x'}, X'$

B. DYNAMIC ACCUMLATORS

Due to the growing volume of the blockchain, the cost of querying the data on the blockchain is becoming larger and larger. At present, dynamic accumulators are attracting more and more researchers' attention. Many accumulators with different characteristics have been proposed. In general, dynamic accumulators are divided into three working directions: RSA accumulators based on strong RSA assumptions, bilinear paired accumulators based on q-SDH assumptions, and accumulators based on anti-collision hashes and hash trees [30].

RSA dynamic accumulator allows dynamic addition or deletion of elements from a given accumulator, and real-time update of existing member witness [31]. The dynamic accumulator implementation process is shown in the following algorithm.

The creation and initialization process of the accumulator is shown in Algorithm 1. Select p, q, and g that meet the conditions to calculate N and $\varphi(n)$.Security parameter 1^{λ} is entered to generate the accumulator administrator's public and private key pairs. The initial empty accumulation value $acc_0 = g \in Z_N$.

The process of adding, deleting, and verifying elements in the dynamic accumulator is shown in Algorithm 2-4. After adding element x, the administrator updates the accumulated value acc and adds the element x to the set X. Deleting element x requires the administrator's private key sk_{acc}. After deletion, update the accumulated value and remove element x from the collection X. The dynamic accumulator accumulates a finite set of elements inside into an accumulative value. For each element in the set, its evidence value W is calculated to prove that the element is inside the accumulator.

IV. PROPOSED PROTOCOL

The authentication scheme is divided into three stages: terminal device identity registration, design of authentication and key agreement protocols in the same domain and cross-domain. The identity registration phase is the same for same domain and cross-domain authenticated IoT devices.

Algorithm 4 Ver Element

Input: cumulative value acc_x,validate elements x_i, witness value W_i, pk_{acc}
Output: *success*, *false*
Compute $acc'_x = W_i^{x_i} \bmod N$
if $acc'_x = acc_x$ **then**
  return *success*
end
else return *false*

TABLE 2. Notation description of authentication and key agreement protocol.

Notation	Description
D_x	The terminal device in domain X
D_y	The terminal device in domain Y
ES_x	The edge server of domain X
ID_x	The identity of the terminal device D_x
$E_{PK}()$	Encryption with public key
$D_s()$	Decryption with private key
$acc()$	Calculate cumulative value function
$Sig()$	Digital signature algorithm
$Ver()$	Signature verification algorithm
$Cert_{D_x}$	The terminal device certificate of domain X
$h()$	Hash function
$A \rightarrow B : \{\}$	Entity A sends a message to entity B
PK	Public key
S	Private key
r	Random number
$\|\|$	Message concatenation
$Cert_{D_{xy}}$	The cross-domain certificate
PK_i	The public key of terminal device i

Table 2 shows the notation description of IoT devices authentication and key agreement protocol.

A. REGISTER

Before the registration and certificate application of same domain terminal devices begin, the public key of the edge server on the local blockchain has been packaged on the Genesis block. A trusted public key is provided for terminal devices identity registration and certificate acquisition.

This section describes the identity registration process of terminal device D in the local blockchain. At this stage, the IoT terminal device D of the local blockchain submits a registration request and encrypts the identity information to the edge server ES. The edge server checks whether the identity information is registered. If it is not registered, it performs the identity registration operation. If it is registered, the registration fails. After successful registration, the terminal device

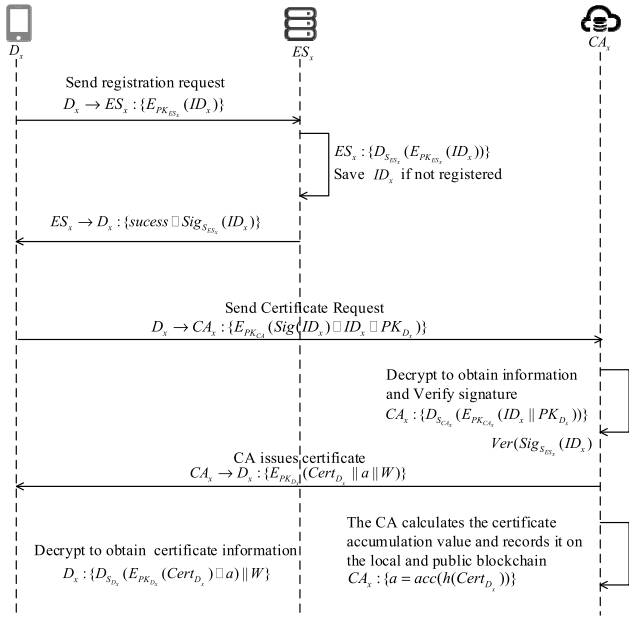


FIGURE 3. Identity registration process.

sends a certificate application to the CA, which requests the edge server to verify the identity information and issue a certificate after verification. Take identity registration and certificate application of terminal device D_x in domain X as an example, and the specific process is shown in Fig. 3.

Step 1: The terminal device D_x belonging to the domain X sends a registration request to the edge server ES_x.

Step 2: When the edge server ES_x receives the registration request, it decrypts request with its own private key S_{ES_x} to obtain the identity of the terminal device D_x. It checks whether the identity is registered, and if not, stores the identity information locally.

Step 3: The edge server ES_x sends the successful registration message and the signed identity information to the terminal device D_x.

Step 4: The terminal device D_x signs ID_x with its own private key and sends $Sig(ID_x)$, ID_x and PK_{D_x} to CA to request a certificate. i.e., D_x sends $E_{PK_{CA}}(Sig(ID_x) \parallel ID_x \parallel PK_{D_x})$ to CA.

Step 5: The CA_x decrypts the message with the private key to obtain the identity information of terminal device D_x and verify the signature. If the verification is passed, the identity information has been successfully registered.

Step 6: The CA_x issues a certificate $Cert_{D_x}$ to terminal device D_x, calculates the hash of certificate $h(Cert_{D_x})$, and packages the certificate to the chain through the consensus algorithm. The cumulative value $a = acc(h(Cert_{D_x}))$ and certificate witness W are generated for the certificate and included in the dynamic accumulator.

Step 7: The terminal device D_x decrypts to obtain the local blockchain digital certificate, certificate cumulative value a

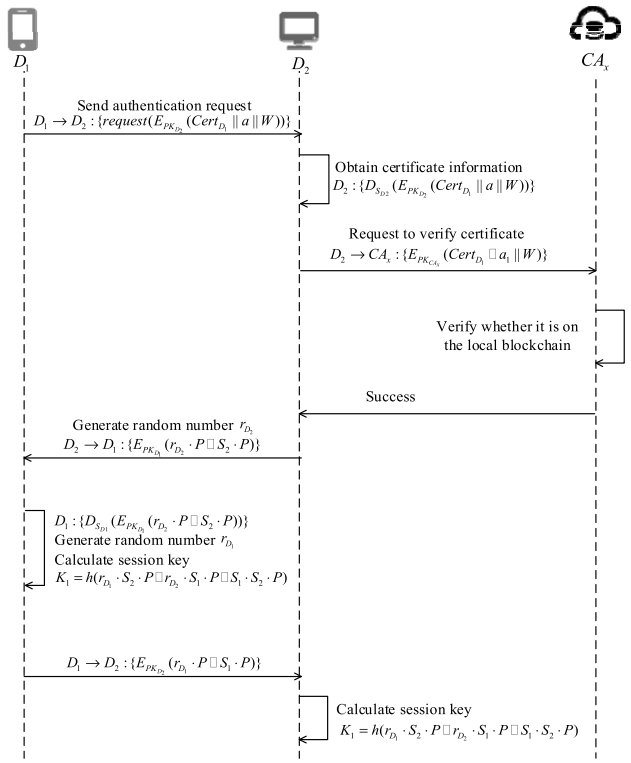


FIGURE 4. Identity authentication and key agreement in the same domain.

and certificate witness W. The information obtained by D_x is used for subsequent identity authentication.

B. AUTHENTICATION AND KEY AGREEMENT IN THE SAME DOMAIN

The terminal device D_1 sends the local digital certificate, certificate cumulative value a and certificate witness W to the terminal device D_2 through public key encryption. As the receiving terminal device D_2 uses the private key to decrypt to obtain the digital certificate, certificate cumulative value a and certificate witness W, and then the terminal device submits it to the CA for review to verify whether the digital certificate is on the blockchain. If it is on the blockchain, it indicates that the certificate verification is passed. After authentication is completed, session key agreement is performed between terminal devices. This is to ensure subsequent secure communication and to improve the speed of encryption and decryption. Take the identity authentication and key agreement of terminal device D_1 and terminal device D_2 in domain X as an example. Describe the process of same domain authentication between terminal devices on the local blockchain. The authentication process is shown in Fig. 4.

Step 1: The terminal device D_1 in the same security domain sends authentication request $E_{PK_{D_2}}(Cert_{D_1}||a||W)$ to terminal device D_2.

Step 2: The terminal device D_2 decrypts with the private key to obtain the certificate $Cert_{D_1}$ and checks the validity of the certificate.

Step 3: D_2 sends the information $E_{PK_{CA_x}}(Cert_{D_1} \| a_1||W)$ to CA to request certificate verification.

Step 4 Upon receiving the digital certificate from the device, the CA_x first checks if the certificate is within its validity period. Then it verifies whether the certificate is on the local blockchain. If the hash of digital certificate value $h(Cert_{D_1})$ is on the blockchain, the authentication success message is returned and sent to D_2. If not, the authentication fails.

Step 5: The terminal device D_2 sends a successful authentic-cation message to D_1. The secure random number r_{D_2} is generated, and the $S_2 \cdot P$ is calculated with the private key S_2. Subsequently $r_{D_2} \cdot P$ and $S_2 \cdot P$ are encrypted and sent to D_1, i.e., D_2 sends $E_{PK_{D_1}}(r_{D_2} \cdot P||S_2 \cdot P)$ to D_1.

Step 6: D_1 decrypts to obtain the $r_{D_2} \cdot P$ and $S_2 \cdot P$. The terminal device D_1 with successful authentication generates the security random number r_{D_1} and calculates the $S_1 \cdot P$ with the private key S_1. Then they are encrypted and sent to the terminal device D_2, i.e., D_1 sends $E_{PKD2}(r_{D_1} \cdot P||S_1 \cdot P)$ to D_2. Finally the terminal device D_1 calculates the session key $K_1 = h(r_{D_1} \cdot S_2 \cdot P \parallel r_{D_2} \cdot S_1 \cdot P \parallel S_1 \cdot S_2 \cdot P)$ for subsequent secure communication with the terminal device D_2.

Step 7: Terminal device D_2 decrypts to obtain the secure random numbers $r_{D_1} \cdot P$ and $S_1 \cdot P$, and it calculates the session key $K_1 = h(r_{D_1} \cdot S_2 \cdot P \parallel r_{D_2} \cdot S_1 \cdot P \parallel S_1 \cdot S_2 \cdot P)$.

C. AUTHENTICATION AND KEY AGREEMENT IN CROSS-DOMAIN

The cross-domain authentication and key agreement process in this section will add new information on the basis of local digital certificates to generate cross-domain certificates. The cross-domain authentication process will re-issue cross-domain certificates as explained below. If the certificate obtained on the local blockchain continues to be used in the cross-domain authentication process, the following problems will occur:

1) It is impossible to distinguish the use scope of the certificate across domains. For example, a terminal device in domain X wants cross-domain access to both domain Y and domain Z. If the same certificate is used for cross-domain authentication, this will result in confusion about cross-domain resource access and failure to recognize the terminal device's cross-domain access rights across different security domains.

2) Cross-domain authentication using certificates on the local blockchain can make cross-domain access less secure. It makes it possible for terminal devices to access other cross-domain resources at will as long as they have a local blockchain domain certificate.

Therefore, cross-domain certificates need to be re-issued when terminal devices are accessed across domains. so that subsequent terminal devices in different security domains have recognizable access rights.

The locally issued certificate is added to the signature of the cross-domain certificate authority CA.The cross-domain certificate is regenerated, the cross-domain certificate hash

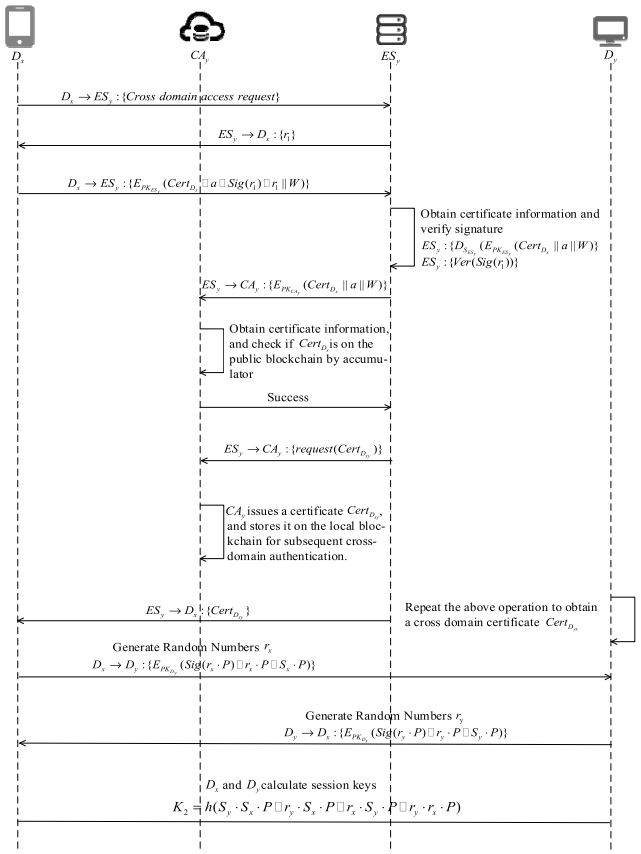


FIGURE 5. Identity authentication and key agreement in cross-domain.

value is calculated, and the smart contract in the public blockchain network is triggered to generate the cumulative value and certificate witness for the certificate, so that the certificate can be verified for subsequent cross-domain access by the terminal device.

In the process of cross-domain authentication and key agreement, *CA*s play the role of a bridge. When end device A in domain X wants to access end device B in domain Y across domains, CA in domain X cannot access the local blockchain in domain Y, and it queries whether the certificate is on the chain through the public blockchain. Different *CA*s in the local blockchain together form the public blockchain.

Due to the existence of different security domains for IoT devices. The terminal device D_x in domain X wants to access terminal device D_y in domain Y across domains need to be authenticated. The session key is jointly negotiated for subsequent secure communication. Cross-domain authentication and key agreement require certificate authority in the public blockchain network to act as a communication bridge. Take the example of terminal device D_x in domain X and terminal device D_y in domain Y.The process of cross-domain authentication and key agreement is shown in Fig. 5.The specific steps are as follows:

Step 1: The terminal device D_x requests cross-domain access to edge server ES_y in domain Y.

Step 2: After receiving the cross-domain access request, the edge server ES_y generates a random number r_1 and sends it to the terminal device D_x.

Step 3: After receiving the random number r_1, the terminal device D_x signs the random number with the private key, and sends the digital certificate $Cert_{D_x}$, certificate cumulative a value and certificate witness W obtained on the local blockchain to ES_y.i.e., D_x sends $E_{PK_{ES_y}}(Cert_{D_x}||a||Sig(r_1)||r_1||W)$ to ES_y.

Step 4: After receiving the message, the edge server ES_y decrypts with the private key to obtain the local blockchain certificate $Cert_{D_x}$, cumulative value a and certificate witness W of the terminal device D_x. It also verifies the validity of the signature and random number r_1. The edge server ES_y sends the digital certificate $Cert_{D_x}$, certificate cumulative a value and certificate witness W to the CA_y to request certificate verification, i.e., ES_y sends $E_{PK_{CA_y}}(Cert_{D_x}||a||W)$ to CA_y.

Step 5: The CA_y obtains the digital certificate, certificate accumulation value a and certificate witness W of the terminal device D_x. The CA_x first checks if the certificate is within its validity period. ThenCA_y node in the public blockchain triggers the dynamic accumulator to query whether the certificate exists. If it exists, the verification passes and returns to ES_y for successful authentication; if it does not exist, it returns for failed authentication.

Step 6: After receiving the successful authentication message, the edge server ES_y requests CA_y to issue a cross-domain certificate $Cert_{D_{xy}}$ for D_x.

Step 7: The CA_y generates a cross-domain certificate $Cert_{D_{xy}}$ to the terminal device D_x.The hash of the cross-domain certificate is calculated and packaged on the local blockchain. This facilitates subsequent cross-domain authentication. And the CA_y sends the cross-domain certificate $Cert_{D_{xy}}$ to ES_y.

Step 8: The ES_y sends the cross-domain certificate $Cert_{D_{xy}}$ to D_x.

Step 9: Repeat the above steps to achieve the reverse authentication. Both D_x and D_y have obtained cross-domain certificates and completed two-way authentication.

Step 10: The terminal devices D_x and D_y that complete two-way cross-domain authentication perform key agreement. The $D_x$generates the security random number r_x, calculates $S_x \cdot P$ with the private key S_x, and sends $r_x \cdot P$ and $S_x \cdot P$ to D_y for key agreement, i.e., $D_x$sends$E_{PK_{D_y}}(Sig(r_x \cdot P) \parallel r_x \cdot P \parallel S_x \cdot P)$ toD_y.

Step 11: The D_y decrypts with private key to obtain $S_x \cdot P$ and $r_x \cdot P$, and verifies the random number $r_x \cdot P$.The D_y generates the secure random number r_y, and uses the private key S_y to calculate $S_y \cdot P$. D_y sends $r_y \cdot P$ and $S_y \cdot P$ to D_x, i.e., D_y sends $E_{PK_{D_x}}(Sig(r_y \cdot P) \parallel r_y \cdot P \parallel S_y \cdot P)$ to D_x. At this point D_y can calculate the session key $K_2 = h(S_y \cdot S_x \cdot P \parallel r_y \cdot S_x \cdot P \parallel r_x \cdot S_y \cdot P \parallel r_y \cdot r_x \cdot P)$

Step 12: The D_x decrypts with private key to obtain $S_y \cdot P$ and $r_y \cdot P$, and calculates the session key $K_2 = h(S_y \cdot S_x \cdot P \parallel r_y \cdot S_x \cdot P \parallel r_x \cdot S_y \cdot P \parallel r_y \cdot r_x \cdot P)$.

V. SECURITY ANALYSIS

In this section, we investigate the security of cross-domain authentication and key agreement protocol and give the security analysis from informal security analysis, formal security proof in the ROR model and formal security analysis with protocol analysis tools.

A. INFORMAL SECURITY ANALYSIS

The informal security analysis describes the security of the scheme in terms of five aspects: certificate security, session key security, forward security, DDoS attacks and man-in-the-middle attacks.

1) CERTIFICATE SECURITY

The same domain and cross-domain authentication schemes designed in this paper both store the digital certificate hash on the blockchain, avoiding the drawback of relying entirely on the centralized *CA* trustworthiness. In addition, if the nodes on the blockchain want to tamper with data, they will pay a huge cost of computing power. To tamper with a certain block data, they must tamper with the entire blockchain, which is not feasible in computing. Therefore, the anti-tampering feature of the blockchain ensures the security of certificates stored on the blockchain. The hash value of certificate is stored on the blockchain, which reduces the storage overhead and effectively alleviates the problem of insufficient storage capacity of the blockchain. Hash functions are collision-resistant, and the probability of a collision on the computation of any two certificate hashes is extremely low. The one-way nature of the hash function makes it impossible to deduce the original input certificate from the hash function. Therefore, hash functions enable blockchain nodes to store certificates anonymously and securely. During same domain and cross-domain authentication, the certificate is sent encrypted with the receiver's public key, and the receiver decrypts it with the private key to obtain the authentication certificate. Under the public key cryptosystem, the public key is public, but the private key is confidential. The public key is calculated from the private key and it is difficult to deduce the private key from the known public key, so the authentication process ensures that the certificate is secure.

2) SESSION KEY SECURITY

In the process of designing session keys in this article, same domain terminal devices D_1 and D_2 send secure random numbers r_{D_1} and r_{D_2} during the agreement of session keys. When a message with a secure random number is received, the recipient of the message verifies that this timestamp is the same as the timestamp sent or that it is a duplicate reception. If the timestamp is repeatedly received, it can be determined that this behavior is a replay attack, so the secure random number can resist replay attacks.

The session key $K_1 = h(r_{D_1} \cdot S_2 \cdot P \parallel S_1 \cdot P \cdot r_{D_2} \parallel S_1 \cdot S_2 \cdot P)$ has unforgeability. When terminal device D_1 sends r_{D_1} encrypted with D_2 public key to D_2, it can only be obtained by decrypting D_2 private key. If there is an attacker, the attacker cannot decrypt and obtain secure random number r_{D_1} without the private key, ensuring the security of the session key.

3) FORWARD SECURITY

Forward security [35] is the inability of an attacker to restore a session key in the event that the private key used to generate the session key is compromised. That is, the previously communicated message remains inaccessible to the attacker. During cross-domain authentication and key agreement, if the private key S_x and private key S_y are inadvertently compromised, the session key K_2 cannot be restored due to the time-sensitive and fresh nature of the secure random numbers r_x and r_y. In addition, the one-way hash function ensures that the session key cannot be easily tampered with. The hash function is not invertible, making it impossible to derive the original input to the hash function by obtaining the session key hash.

4) DDOS ATTACK

Distributed Denial of Service Attacks (DDoS Attacks) are designed to make the target unable to provide services properly by sending a large number of requests to the target. The goal of a DDoS attack is to overload the target system so that it is unable to process normal user requests. Blockchain has distributed nature. In a blockchain network, each node holds complete information about the ledger. Even if less than one-third of the total number of nodes in the network fail, the normal use of the ledger will not be affected. This means that even if some of the nodes are not working, the server will still be able to obtain authentication information from the blockchain network and realize cross-domain authentication. Once the attacked node is back to normal, it can obtain complete ledger information from other nodes and become a normal node with cross-domain authentication again.

5) MAN-IN-THE-MIDDLE ATTACK

A man-in-the-middle attack is when an attacker inserts himself as a ''middleman'' between two parties to obtain sensitive information or tamper with the content of the communication. Suppose the attacker intercepts the random number N sent by the RA to the device, but the attacker cannot sign the random number due to the lack of the device's private key. Even if the attacker tries to sign the random number with its own private key, it needs to send its own certificate to the RA, which will not be able to authenticate the attacker's certificate via the distributed ledger. The authentication will succeed only if the device signs the random number with its own private key and sends the certificate information to the RA.

B. FORMAL SECURITY ANALYSIS IN ROR MODEL

In this subsection, we give a formal security proof for the proposed protocol in the ROR model. The ROR model is a classical security model for proving the security of authentication and key agreement protocols [37], [38].

In ROR model, the ith protocol instance of U is modeled as an oracle $\prod_U^i$. There is an adversary A who can control the communication channel and his/her purpose is to break the authentication or gets the session key. There is a simulator S simulates the protocol and A interacts with the oracles $\prod_U^i$to gets the information he/she wants. As mentioned in [39], the attacks of the adversary are simulated as the queries to the oracles in the protocol in ROR models which are Execute(U1,U2), Send ($\prod_U^i$, m), Corrupt(U), Reveal ($\prod_U^i$) and Test($\prod_U^i$). We do not repeat the definitions here.

Definition 1: Elliptic curve Diffie-Hellman problem (ECDH problem). Let G is an elliptic curve additive group over finite field F_p, and P is a generator of G and the order of P is q where q is a large prime. aP and bP are two random point of G where a and b are unknown. There is no adversary A can compute abP in polynomial time. Let $Succeed_G^{ECDH}(A)$ be the probability that A computes abP mentioned above, then we know for every probabilistic polynomial adversary A, $Succeed_G^{ECDH}(A)$ is negligible.

Theorem 1: Let G is an elliptic curve additive group over finite field F_p, and P is a generator of G. P is the protocol proposed in this paper. For any probabilistic polynomial-time A with less than q_e times Execute query, less than q_s times Send query and less than q_h times Hash query, then the probability that A break the semantic security of P is as follows:

$Adv_G^P(A) \leq$*Proof.* The security proof of the protocol is completed by the games between adversary A and the simulator S. S simulates the protocol and answers the queries from the adversary A. Let G_i denotes the ith game between A and S, $Succ_i$ denotes the event that A successfully distinguishes a random value or the session key returned from the Test query. Let $Diff_i = \Pr[Succ_i] - \Pr[Succ_{i-1}]$ denotes the probability difference between ith event and i-1th event.

Game G_0 This game simulates the real attack to the real protocol, so we have:

$$Adv_G^P(A) = |2\Pr[Succ_0] - 1|$$

Game G_1 This game is similar to G_0, the difference is that S runs the game in the random oracle model which means we answer the Hash query with random value. From the definition of the random oracle model, we have:

$$Diff_1 = \Pr[Succ_1] - \Pr[Succ_0] = 0$$

Game G_2 In this game, we exclude the collision between different protocol instances, i.e., S or A chooses the same random value in different protocol instance. Because if the collision on the protocol instances happens, then A can successfully guess the answer from the Test query and distinguish between the session key and the random value(A can choose one of the session asks a Reveal query and get the session key). We also exclude the collision on the Hash functions, so according to the birthday paradox, we have:

$$Diff_2 = \frac{(q_e + q_s)^2 + q_h^2}{2p}$$

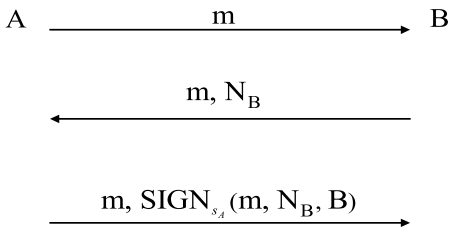


FIGURE 6. The authenticator in BCK.

Game G_3 In this game, we use the authenticator proposed in the BCK security model [39] to instead the authentication process of the proposed protocol in this paper. Actually, the authentication process of the proposed protocol is similar to that of the authenticator in BCK model. The authenticator in BCK is as follows:

From Fig.6 we can see the authentication of the proposed protocol is the same as the authenticator in BCK in essence. In every authentication process we use a random value and the signature of the message which means the authentication of the message is secure. If adversary A can break the authentication, then it means an algorithm can break the security of the authenticator in BCK by calling A as a subroutine. However, the security proof of the signature-based authenticator was already given in [39]. So we have:

$$Diff_3 = Adv_G^{authenticator}(A) \leq Succeed_G^{ECDH}(A)$$

Game G_4 In this game, we consider an adversary with strong ability which means he/she can corrupt one of participants, then he/she takes a Test query to the simulator S. If the protocol can provide semantic security in this case, then the protocol can also provide semantic security in other cases since the adversary has weaker ability. In order to prove the conclusion we choose a random ECDH tuple $< P, aP, bP >$ to instead the public key of D_x and the random parameter r_yP chosen by D_y in the protocol. In such case, we need to prove the simulation of the protocol is correct after the substitution. Since the public key of D_x is replaced by aPso we do not know the parameter a of aP, then we cannot answer the Reveal query. The reason is that we cannot compute the real session key without the parameter a. In this case, we first check whether there is a record $< S_y \cdot aP \parallel r_y \cdot aP \parallel r_x \cdot S_y \cdot P \parallel r_y \cdot r_x \cdot P, h_i >$in the Hash list, if there is a record, we let h_i be the session key of this session and answer the Reveal query with h_i. Otherwise, we choose a random value h_i' and put $< S_y \cdot aP \cdot P \parallel r_y \cdot aP \parallel r_x \cdot S_y \cdot P \parallel r_y \cdot r_x \cdot P, h_i' >$ in the Hash list and let h_i' be the session key of this session and answer the Reveal query with h_i'. Now, the simulator S can simulate the protocol correctly. In such case, if A can distinguish the difference between the session key of the test session and a random value, then it means A must have ask a hash query with $< S_y \cdot aP \parallel b \cdot aP \parallel r_x \cdot S_y \cdot P \parallel b \cdot r_x \cdot P >$, then the simulator S can check the hash record and get the value $a \cdot bP$. So it means the ECDH problem can be broken by using A as a subroutine. Suppose the probability that the test session that

A chose is exactly the one that the we imbed the ECDH tuple is $1/(q_e + q_s)$ and the probability that the matching session of these two session is also the same is $1/(q_e + q_s)$, then we have:

$$Diff_4 = (q_e + q_s)^2 q_h \cdot Succeed_G^{ECDH}(A)$$

To sum up, we have:

$$\begin{aligned} Adv_G^P(A) &= |2\Pr[Succ_0] - 1| \\ &= |2\Pr[Succ_0] - 1 + 2\Pr[Succ_4] - 2\Pr[Succ_4]| \\ &= |2\Pr[Succ_4] - 1 + 2(\Pr[Succ_0] - \Pr[Succ_4])| \\ &\le |2\Pr[Succ_4] - 1| + 2\sum_{i=1}^{4} Diff_i \end{aligned}$$

In Game G_4, the protocol is already run in the random oracle model, so $\Pr[Succ_4] = \frac{1}{2}$. Now we can conclude the Theorem 1 as follows:

$$Adv_G^P(A) \le \frac{(q_e + q_s)^2 + q_h^2}{p} + 2(q_e + q_s)^2 q_h \cdot Succeed_G^{ECDH}(A)$$

C. FORMAL SECURITY ANALYSIS WITH PROTOCOL ANALYSIS TOOLS

Scyther is an important tool for verifying protocol security. The verification process uses an unlimited number of sessions and random numbers to verify whether the protocol is secure. Scyther tool provides graphical operation interface with good human-computer interaction, and can graphically output protocol attacks. Scyther tool can add adversary models under strong security models such as long-term key disclosure, session key disclosure, random number disclosure, and state disclosure by checking [36].

This section uses the Scyther tool to experiment with security analysis of authentication and key agreement protocols in the same domain and cross-domain.

1) SAME DOMAIN

The scyther tool analyses the process of same domain authentication and key agreement protocols, specifically involving the three entities D_A, D_B and CA within the same security domain. The simulation results show that the declared secret, alive, weakagree, niagree, and nisynch attributes have not found attacks outside or inside the state space boundaries. The experimental results are shown in Fig. 7.

Description of the same domain authentication protocol: defines three entities D_A, D_B, and CA, and the specific process is shown in Fig.8.

Testing key agreement protocol security, as depicted in Fig. 9. The subsequent cross-domain authentication and key agreement protocol experiment process is similar to the security analysis experiment of the same domain authentication and key agreement process. The specific code of the experiment will not be discussed in detail.

Scyther results : verify

Claim				Status		Comme
tongyu	DA	tongyu,DA1	Secret {Cert}pk(DB)	Ok	Verified	No attacks.
		tongyu,DA2	Secret {rDB,exp(P,S2)}pk(DA)	Ok	Verified	No attacks.
		tongyu,DA3	Secret {rDA,exp(P,S1)}pk(DB)	Ok	Verified	No attacks.
		tongyu,DA4	Secret K1	Ok	Verified	No attacks.
		tongyu,DA5	Alive	Ok	Verified	No attacks.
		tongyu,DA6	Weakagree	Ok	Verified	No attacks.
		tongyu,DA7	Niagree	Ok	Verified	No attacks.
		tongyu,DA8	Nisynch	Ok	Verified	No attacks.
	DB	tongyu,DB1	Secret {rDB,exp(P,S2)}pk(DA)	Ok	Verified	No attacks.
		tongyu,DB2	Secret {exp(P,S1)}pk(DB)	Ok	Verified	No attacks.
		tongyu,DB3	Secret K1	Ok	Verified	No attacks.
		tongyu,DB4	Alive	Ok	Verified	No attacks.
		tongyu,DB5	Weakagree	Ok	Verified	No attacks.
		tongyu,DB6	Niagree	Ok	Verified	No attacks.
		tongyu,DB7	Nisynch	Ok	Verified	No attacks.
	CA	tongyu,CA1	Secret {Cert,a1}pk(CA)	Ok	Verified	No attacks.
		tongyu,CA2	Secret K1	Ok	Verified	No attacks.
		tongyu,CA3	Alive	Ok	Verified	No attacks.
		tongyu,CA4	Weakagree	Ok	Verified	No attacks.
		tongyu,CA5	Niagree	Ok	Verified	No attacks.
Done.		tongyu,CA6	Nisynch	Ok	Verified	No attacks.

FIGURE 7. Same domain authentication and key agreement security verification results.

2) CROSS-DOMAIN

The Scyther tool analyses the process of cross-domain authentication and key agreement protocols, specifically involving the four entities D_x, D_y, ES_x, and ES_y within the different security domains. The simulation results show that the declared Secret, Alive, Weakagree, Niagree, and Nisynch attributes in D_x and D_y communication are not found within the limits. The Secret, Alive, Weakagree, Niagree and Nisynch attributes declared by ES_x and ES_y are not found to be attacked outside or inside the state space boundaries. The experimental results are shown in Fig. 10.

VI. BLOCKCHAIN IMPLEMENTATION

A. PLATFORM INTRODUCTION

The FISCO BCOS platform is an open source blockchain platform launched by the Financial Blockchain Platform Consortium of China (FISCO) [40]. It realizes high-performance transaction processing capability, which can support thousands to tens of thousands of transactions per second. In terms of guaranteeing data security and reliability FISCO BCOS adopts PBFT (Improved Byzantine Fault Tolerance) consensus algorithm, which ensures that more than 2/3 of the nodes in the network are honest. It also provides rich smart contract support and supports alliance chain expansion.

WeCross is a cross-chain solution launched by Tencent's blockchain team, aiming to solve the problem of blockchain

```
// Defining text types and hash functions
usertype text;
hashfunction H, exp;
const SUCC:text;

// Define the protocol tongyu, which contains three roles: DA, DB and CA.
protocol tongyu(DA, DB, CA) {

  // DA Role Definition
  role DA {
        fresh Cert: Nonce;
        Send_1(DA, DB, {Cert}pk(DB));
        fresh rDA, P,S1: Nonce;
        var rDB, S2: Nonce;
        var K1: SessionKey;
        recv_4(DB, DA, {rDB, exp(P, S2)}pk(DA));
        send_5(DA, DB, {rDA, exp(P, S1)}pk(DB));
        // Verify that the session key K1 matches
        match(K1, H(exp(rDA, exp(P, S2)), exp(rDB, exp(P, S1)), exp(S2, exp(P, S1))));
              claim(DA, Secret, {Cert}pk(DB));
              claim(DA, Secret, {rDB, exp(P, S2)}pk(DA));
              claim(DA, Secret, {rDA, exp(P, S1)}pk(DB));
                    claim(DA, Secret, K1);
                    claim(DA, Alive);
                    claim(DA, Weakagree);
                    claim(DA, Niagree);
                    claim(DA, Nisynch);
          // Authentication phase end
  }

  // DB Role Definition
  role DB {

    fresh a1: Nonce;
    var Cert: Nonce;
    // Authentication phase begin
    recv_1(DA, DB, {Cert}pk(DB));
    send_2(DB, CA, {Cert, a1}pk(CA));
    var rDA, S1: Nonce;
    fresh rDB, S2, P: Nonce;
    var K1: SessionKey;
    recv_3(CA, DB, SUCC);
    send_4(DB, DA, {rDB, exp(P, S2)}pk(DA));
    recv_5(DA, DB, {rDA, exp(P, S1)}pk(DB));
    // Verify that the session key K1 matches
       match(K1, H(exp(rDA, exp(P, S2)), exp(rDB, exp(P, S1)), exp(S2, exp(P, S1))));
    send_6(DB, CA, K1);

                  claim(DB, Secret, {rDB, exp(P, S2)}pk(DA));
                  claim(DB, Secret, {exp(P, S1)}pk(DB));
                  claim(DB, Secret, K1);
                  claim(DB, Alive);
                  claim(DB, Weakagree);
                  claim(DB, Niagree);
                  claim(DB, Nisynch);
    // Authentication phase end
  }

  // CA Role Definition
  role CA {
      //
      fresh Cert, a1: Nonce;
      recv_2(DB, CA, {Cert, a1}pk(CA));
      send_3(CA, DB, SUCC);
    // Authentication phase end
  }
}
```

FIGURE 8. Security experiment of authentication protocol in the same domain.

cross-chain interconnection. It is a lightweight cross-chain gateway and framework that enables interconnection and

```
// Define a session key variable K1
var K1: SessionKey;
// The CA receives a sixth message from the DB
containing the session key K1
recv_6(DB, CA, K1);

    claim(CA, Secret, {Cert, a1}pk(CA));
    claim(CA, Secret, K1);
    claim(CA, Alive);
    claim(CA, Weakagree);
    claim(CA, Niagree);
    claim(CA, Nisynch);
```

FIGURE 9. Security experiment of key agreement protocol in the same domain.

Scyther results : verify

Claim				Status		Comments
kuayu	Dx	kuayu,Dx1	Secret sign	Ok		No attacks within bounds.
		kuayu,Dx2	Secret signx	Ok		No attacks within bounds.
		kuayu,Dx3	Secret K2	Ok		No attacks within bounds.
		kuayu,Dx4	Alive	Ok		No attacks within bounds.
		kuayu,Dx5	Weakagree	Ok		No attacks within bounds.
		kuayu,Dx6	Niagree	Ok		No attacks within bounds.
		kuayu,Dx7	Nisynch	Ok		No attacks within bounds.
	Dy	kuayu,Dy1	Secret signy	Ok		No attacks within bounds.
		kuayu,Dy2	Secret K2	Ok		No attacks within bounds.
		kuayu,Dy3	Alive	Ok		No attacks within bounds.
		kuayu,Dy4	Weakagree	Ok		No attacks within bounds.
		kuayu,Dy5	Niagree	Ok		No attacks within bounds.
		kuayu,Dy6	Nisynch	Ok		No attacks within bounds.
	ESx	kuayu,ESx1	Alive	Ok	Verified	No attacks.
		kuayu,ESx2	Weakagree	Ok	Verified	No attacks.
		kuayu,ESx3	Niagree	Ok	Verified	No attacks.
		kuayu,ESx4	Nisynch	Ok	Verified	No attacks.
	ESy	kuayu,ESy1	Secret r1	Ok	Verified	No attacks.
		kuayu,ESy2	Secret Certxy	Ok	Verified	No attacks.
		kuayu,ESy3	Alive	Ok	Verified	No attacks.
		kuayu,ESy4	Weakagree	Ok	Verified	No attacks.
		kuayu,ESy5	Niagree	Ok	Verified	No attacks.
Done.		kuayu,ESy6	Nisynch	Ok	Verified	No attacks.

FIGURE 10. Cross-domain authentication and key agreement security verification results.

data transfer between different blockchains. Meanwhile, WeCross is built on a lightweight cross-chain gateway, which makes integration and usage simple and efficient. It also supports many different blockchain platforms, including FISCO BCOS, Fabric, Ethereum, and so on. Its cross-chain interaction protocols and data formats are compatible with mainstream blockchain platforms, facilitating integration with other blockchain systems.

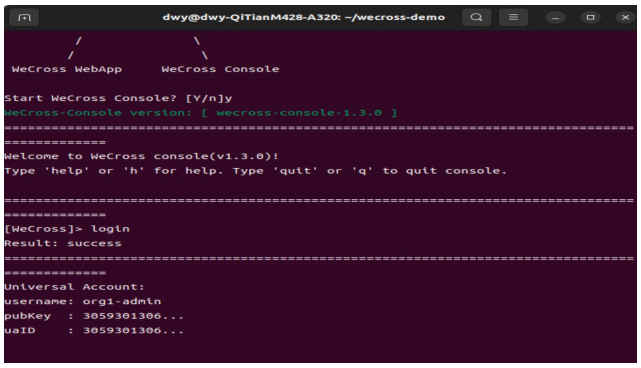


FIGURE 11. Show environment configuration success and login success.

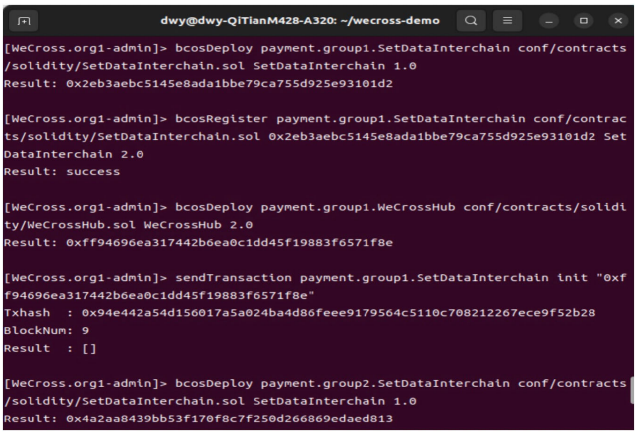


FIGURE 12. Successful contract deployment.

B. IMPLEMENTATION PLAN

In this paper, the platform used to build the blockchain based on Ubuntu 22.04.1 version of the system is fiscobcos version 2.0. Two blockchains are built on the platform, group1 and group2, where group1 is the public blockchain and group2 is the local blockchain. The consensus algorithm uses the PBFT algorithm that comes with the platform. At the same time, we use the wecross v1.3.0 cross-chain platform to realize the deployment of smart contracts on the blockchain and the interaction between the blockchain chains.

Group1, as the public blockchain, is responsible for storing and querying the certificate hash values transmitted by all local blockchains. Group 2, as a local blockchain, can store and query the hash value locally and upload the local hash value to the public blockchain. At the same time, it realizes the querying of other hash values on the public blockchain through smart contracts (the blockchain 1 and blockchain 2 in the following are group1 and group2).

C. DEPLOYMENT PROCESS

1) INSTALL AND CONFIGURE THE ENVIROMENT

Install wecross and configure the relevant runtime environment. In this article, we have configured javaJDK version 1.8.0_362, openssl version 3.0.2 and mysql version 8.0.33. You can also sudo apt-get install -y openssl curl expect tree fontconfig command to configure the environment. After configuring the environment, download the demo from the wecross website. After the download is complete, run the command cd ~/wecross-demo to enter the download directory of the demo, and execute bash clear.sh to clean up the old installation environment. Run bash build_cross_groups.sh to deploy the scripts. When all the script files are successfully deployed, a 4-node blockchain with two groups will be built directly from the official configuration file. The consensus mechanism between the nodes uses the officially configured PBFT (Practical Byzantine Fault Tolerance) consensus algorithm. Fig.11 shows successful environment configuration and successful login.

2) IMPORTING SMART CONTRACTS

After successful configuration, you can directly enter the wecross command console. Enter login to log in to the console for contract deployment operations. Firstly, place the written smart contract in the contract file of the console in the directory of/wecross-demo/WeCross-Console/conf/contracts/policy.

3) DEPLOYING CONTRACTS ON THE BLOCKCHAIN

In the console, first deploy the contract in blockchain 1, and the contract deployment command is bcosDeploy. Firstly, deploy the SetDataInterchain contract, and upon successful deployment, the address of the contract will be returned. Use the bcosRegister command to register the SetDataInterchain contract as a cross chain resource. This way, we can perform cross chain operations through this contract. The contract address that needs to be filled in when calling the bcosRegister command is the address obtained from the previous deployment of the contract. If the result returns success, it indicates successful registration. Continue deploying the WeCrossHub contract. After the contract deployment is completed, you need to use the sendTransaction command to call the init function in the SetDataInterchain contract to associate the addresses of the WeCrossHub contract. Displaying BlockNum indicates that the association was successful, enabling the SetDataInterchain contract to call the cross-chain function in the WeCrossHub contract. Next, MDAC, VFS, and TC contracts can be deployed in blockchain 1 using the same method. Deploy DAC, TCA, WeCrossHub, and SetDataInterchain contracts in blockchain 2, and the smart contracts on both blockchains and blockchains are successfully deployed. Fig.12 shows a successful smart contract deployment on the blockchain.

D. INTRODUCTION TO SMART CONTRACT FUNCTIONALITIES

The relevant smart contracts deployed on the blockchain1 are MDAC, VFS, TC, WeCrossHub and SetDataInterchain.

Among them, WeCrossHub and SetDataInterchain are contracts shared by the public blockchain and local blockchains, which are responsible for cross-chain interactions between the public blockchain and local blockchains.

The SetDataInterchain contract is responsible for the interaction between the child chain and the public blockchain. There is a setDataInterchainInvoke function in the contract which is responsible for inputting information. Five parameters need to be filled in, which are the name of the contract on the target chain, the name of the function on the contract, the parameters of the function, the contract on the chain where the callback function is located, and the method name of the callback function. Then WeCrossHub is called to make the cross-chain call.

The SetDataInterchain contract part of the code is shown in Algorithm 5. This function is used to make interchain calls on the blockchain and pass data and callback information.

Algorithm 5 SetDataInterchainInvoke

```
Input: path,method,data, callbackPath,
callbackMethod
Output:
// This function is used to make cross-chain calls on
  the
blockchain and pass data and callback information.
string memory _path,
string memory _method,
string memory _data,
string memory _callbackPath,
string memory _callbackMethod
// Create a string array 'args' with one element to
  store the
parameter data passed to the target smart contract.
public returns (string memory)
    string[] memory args = new string[](1);
    args[0] = _data;
    return hub.interchainInvoke(
    _path,
_method,
    args,
    _callbackPath,
_callbackMethod
);
```

TC contract is responsible for receiving the hash value from the SetDataInterchain contract on the local blockchain and storing the hash value for MDAC to call.VFS contract is responsible for receiving the hash value transmitted from the SetDataInterchain contract on the local blockchain and storing the hash value for MDAC to call. And call the query function in the public blockchain through verifyCertificateFM function to query whether the hash value is in the public blockchain and return the query result. The MDAC contract is responsible for storing and querying the hash value uploaded by the child chain. They are executed by registerCertificateFromTC function and verifyCertificate function respectively. In registerCertificate-FromTC function, it will directly receive the hash value from the TC contract as a parameter, and there will be a require to deposit the hash value to determine whether the hash value is in the chain. If the hash value already exists, there will be a corresponding message. If the hash value is stored successfully, the hash value will be stored in the accumulator and added to the chain. Use verifyCertificate function to query the hash value can be directly in the accumulator to query the hash value. If the certificate exists then return ''certificate in the chain'', otherwise return ''certificate is not in the chain''. The MDAC contract part of the code is shown in Algorithm 6.

Algorithm 6 RegisterCertificateFromTC

```
Input: publicKeyHash
Output: whether or not it is stored on the chain
// Get the hash value created by the set function in the
  TC
contract
bytes32 publicKeyHash = tcContract.get();
require (publicKeyHash!=bytes32(0), ''No certificate
  hash
    found in TC contract'');
require(certificates[publicKeyHash].publicKeyHash==
    bytes32(0), ''Certificate already registered'');
// The certificate hash is associated to the certificate
certificates[publicKeyHash]=Certificate(publicKeyHash);
    addToAccumulator(publicKeyHash);
// Certificate Enrollment event is triggered
emit CertificateRegistered(publicKeyHash);
```

The relevant smart contracts deployed on the local blockchain are DAC, TCA, WeCrossHub and SetDataInterchain. The DAC contract is responsible for entering and querying certificates on the local chain. The registerCertificate function stores the name and hash value of the certificate into the accumulator. The accumulator is called with verifyCertificate function to query the certificate. If the certificate exists, it returns ''certificate is in the chain'', otherwise it returns ''certificate is not in the chain''. The DAC contract part of the code is shown in Algorithm 7.

Algorithm 7 VerifyCertificate

```
Input: publicKeyHash
Output: whether the certificate is on the chain
// Call the function to verify that the hash value of the
certificate is in the accumulator
bool exists = checkInAccumulator(publicKeyHash);
if exists then
    return certificate is in the chain
else
    return certificate is not in the chain
```

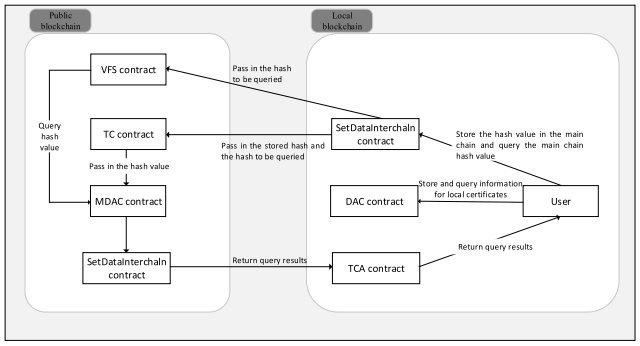


FIGURE 13. Operation process flow diagram.

The TAC contract is responsible for getting the results returned by the SetDataInterchain contract on the public blockchain. The result information is queried through the get function.

E. OPERATION PROCESS

Fig. 13 illustrates the specific operational process.

Step 1: Call registerCertificate function in the DAC contract in blockchain 2 (local blockchain) to input the certificate name and hash value and register it on the chain.

Step 2: Check if the certificate is in the chain by verifyCertificate function. If the certificate is in the chain, then return ''certificate is in the chain''.

Step 3: Call the SetDataInterchain contract to pass the hash value of the certificate stored in the local blockchain to the contract TC in the public blockchain.

Step 4: The contract TC in blockchain 1 (public blockchain) receives the hash value and stores it, and queries whether the hash value has been successfully imported through the get function in the TC. If the hash value is successfully imported, the encrypted information of the hash value will be displayed.

Step 5: Call the registerCertificateFromTC function in the MDAC contract in the public blockchain to register the hash value in the TC contract directly on the chain. Call verifyCertificate function in MDAC contract of public blockchain to check whether the certificate is stored successfully. If successful, it will return ''certificate in chain'', then the hash value of the certificate in the local blockchain will be stored in the public blockchain.

Step 6: Blockchain 2 (local blockchain) to blockchain 1 (public blockchain) to launch the hash value query. The SetDataInterchain cross-chain contract calls the VFS contract on blockchain 1 (the public blockchain) to query the hash value of the certificate stored on the public blockchain. The VFS contract calls the verifyCertificateFM function to perform a query on the hash value on blockchain 1 (the public blockchain). If the hash value is in blockchain 1 (public blockchain), the return result is passed back through SetDataInterchain in blockchain 1 (public blockchain) to the TCA contract in blockchain 2 (child chain).

TABLE 3. Algorithm running time.

Algorithm Category	Algorithm Name	Average Running Time
T_h	Hash algorithm	0.001ms
T_p	Asymmetric encryption and decryption algorithm	4.050ms
T_s	Digital signature and verification algorithm	2.100ms
T_e	Bilinear pairing algorithm	11.32ms
T_d	Dot multiplication	0.220ms
T_g	Exponential operation	6.532ms

Step 7: Blockchain 2 (child chain) calls the get function of TCA contract summary to check the return result.

VII. PERFORMANCE ANALYSIS

This section focuses on the performance analysis of the cross-domain authentication and key agreement protocol for IoT devices in the previous section, including computing overhead, communication overhead, experiment comparison and performance experiment. The desktop computer used in the experiment is configured with an Intel Core i5-10400 CPU@3.00GHZ processor and 16GB of RAM, and the operating system is Windows 10 (64bit).

A. COMPUTING OVERHEAD

This section uses the following notation to indicate the name and running time of each algorithm, as shown in Table 3. IoT devices use ECDSA (Elliptic Curve Digital Signature Algorithm) signatures, which involve two point multiplication operations on the elliptic curve. An average of 2.1ms is consumed for a single digital signature operation. This amount of computation is feasible for ordinary devices.

The authentication process requires 10 public key encryption and decryption operations, which results in a relatively high computational cost [28]. In [21], the use of multiple bilinear operations further increases the cost compared to what is presented in this paper. The method of certificate verification is carried out by comparing the certificate hash submitted by the cross-domain user with the certificate hash stored on the blockchain. This approach increases the query time when searching on the blockchain, as it necessitates traversing the entire blockchain. In this paper, however, we employ a dynamic accumulator in the authentication process to reduce the certificate query time complexity. While [27] designed a master-slave blockchain architecture, the computational cost was higher than the cross-domain authentication scheme proposed in this paper. Table 4 illustrates the comparative analysis of the computational costs. Fig. 14 and Table 4 show the comparison of calculation costs.

Therefore, this scheme has good computational overhead. The authentication process information is sent encrypted with the other party's public key, reducing the chance of malicious attackers gaining access to identity information. During the

TABLE 4. Computing overhead comparison.

Scheme	Calculations Total	Running Time
G. Cheng et al. [28]	$10T_p + 2T_s + T_h$	44.701ms
S.Y. Guo et al. [27]	$8T_p + 4T_s + T_h$	40.801ms
C. Yuan et al. [21]	$2T_g + 3T_s + 3T_h + T_d$	29.163ms
our scheme	$4T_p + T_s$	18.3ms

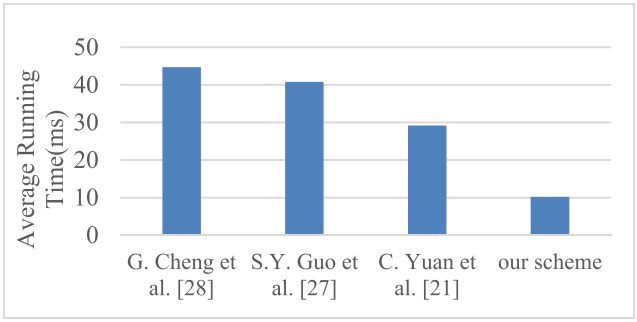


FIGURE 14. Comparison of computing overhead.

TABLE 5. Parameter lengths.

Parameters	Description	Size(bites)
G	Bit length of an element in G	512
ID	Bit length of an identity	256
T	Bit length of a time-stamp	32
r	Bit length of a random number	256
h	Bit length of a hash function	256
PK	Bit length of the public key cipher	1024
SK	Bit length of the private key cipher	256
e	Bit length of the Bilinear mapping	1024

TABLE 6. Comparison of bandwidth consumption.

Scheme	Bandwidth Consumption(bites)
G. Cheng et al. [28]	6080
S.Y. Guo et al. [27]	5280
C. Yuan et al. [21]	4544
Our scheme	3584

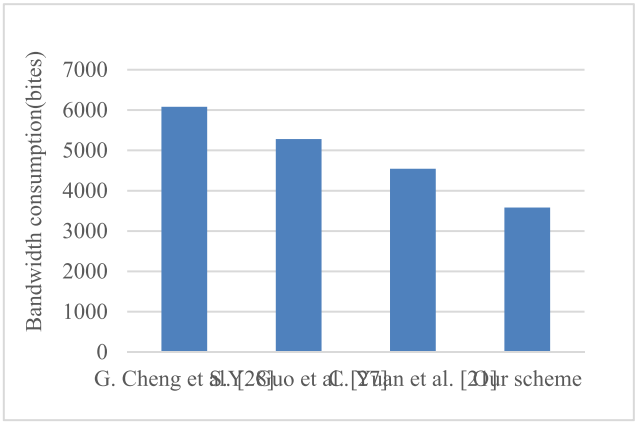


FIGURE 15. Comparison of bandwidth overhead.

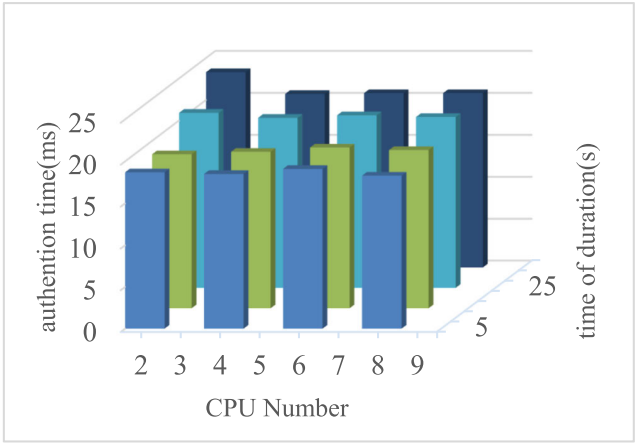


FIGURE 16. Time of cross-domain authentication.

authentication process, this scheme improves the query efficiency of the certificate through dynamic accumulator.

B. COMMUNICATION OVERHEAD

In order to better analyze our cross-domain authentication and key agreement protocols, we have listed the bit sizes of different parameters in Table 5. In Table 6 and Fig.15, we compared the communication costs of our protocol with other cross-domain authentication protocols, and it can be seen that our protocol is more efficient than other protocols.

C. PERFORMANCE EXPERIMENT

In this section, simulation experiments are conducted to test the concurrent performance of cross-domain authentication. The test program sent cross-domain authentication requests in 1s, 10s, 20s and 30s. VMs were configured with two CPUs, four CPUs, six CPUs and eight CPUs to invoke to achieve cross-domain authentication. In the experimental tests, the single authentication time is shown in Fig. 16. The experimental results show that the single authentication time is about 18 to 23ms and our cross-domain authentication scheme has good stability and high efficiency.

The traditional way of querying data on a blockchain requires traversing the entire blockchain. As the blockchain grows in size, querying becomes inefficient. In this paper, by constructing a dynamic accumulator, the way of traversing the data in the blockchain is replaced by proving that the members are in the accumulator. This enables the time complexity of the query to be reduced from O(n) to O(1), reducing query time consumption and improving the efficiency of cross-domain authentication. As the number of members in the accumulator increases, the time complexity of adding or removing members from the dynamic accumulator does not increase. It is efficient at adding and removing members. Although dynamic accumulators are time consuming

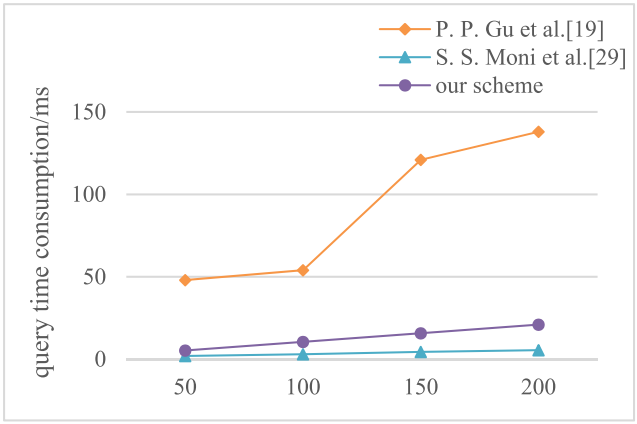


FIGURE 17. Average certificate query time.

to create, the process is a one-off and can be done in the background.

In this paper, the dynamic accumulator uses the strong RSA assumption in cryptography to ensure the security of the dynamic accumulator scheme. It is a positive integer with a length of 1024, which is the product of the sum generated by two Fermat primality detection algorithms. First, 3000 blockchain cross-domain certificates are created in batch as a test data set. It needs to repeat the experiment and average the data for five times, which can reduce errors and avoid contingency. Fig. 17 shows the average time spent querying 50,100,150,200 certificates.

The average query time of [19] grows linearly with the number of certificates as shown in the Fig. 17. The time consumption of this paper does not fluctuate much with the number of certificates, so it is suitable for edge computing environments with a large number of terminal devices. Although the query speed of this paper lags behind that of [29], there is a 3% probability of misjudgment in [29]. If it determines that a certificate exists, it may not actually exist, leading to privacy leaks and security issues.

D. SYSTEM ARCHITECTURE ANALYSIS

Multi-layer blockchain provides distributed authentication with the help of blockchain technology. Different security domains correspond to a local blockchain to establish a local collaborative trust network. The data processed by a single device is distributed to multiple edge servers for collaborative processing, which ensures the consistency and security of decentralized IoT systems. The multi-tier blockchain is composed of a public blockchain and a local blockchain. Both public or local blockchains are based on federated chain composition. External nodes can join only after getting approval from the audit to improve the security of the system architecture.

In the local blockchain, it supports the authentication of terminal devices in the same domain, manages each node in the IoT, and secures the data in the domain. In the public blockchain, establish communication channels for edge node servers in different domains and maintain the trust relationship between CAs in different domains. The hash value of the digital certificate is stored in the public blockchain. It improves the reliability of certificates and reduces the storage pressure on the public blockchain. The authentication process is combined with dynamic cryptographic accumulator technology. It solves the problem of low efficiency of data query on the chain due to the increase of blockchain volume and improves the efficiency of certificate verification of the authentication process. When terminal devices belonging to different security domains need authentication communication, the public blockchain network is used as a communication bridge to ensure the secure sharing of information across domains. The architecture can effectively utilize the computing power of different infrastructures and enhance the scalability of the system. It can also effectively relieve the pressure of individual blockchain storage and reduce the burden of blockchain queries.

But multi-layer blockchain architectures also have limitations. It does not have a consensus mechanism suitable for this architecture. Consensus mechanisms are algorithms for reaching distributed consensus on blockchain transactions. This architecture is composed of local blockchain and public blockchain. There are respective consensus mechanisms in the blockchain and there is no overall consensus mechanism implemented for the multilayer blockchain architecture yet.

VIII. CONCLUSION

In the edge computing environment, this scheme proposes a cross-domain authentication and key agreement protocol based on a multi-layer blockchain. Cross-domain authentication of IoT devices with different security domains is achieved. A multi-layer blockchain architecture is designed, consisting of a local blockchain and a public blockchain. Dynamic accumulator is introduced to solve the problem of inefficient certificate lookups. Next we conducted performance and security analysis, and the results showed that the protocol is well feasible and efficient, and more adaptable with low performance devices. Further in-depth research on the underlying blockchain technology is needed in the future to make blockchain technology an important tool for identity authentication and key agreement.

REFERENCES

[1] *The Mobile Economy 2020*, GSMA Assoc., London, U.K., 2019. [Online]. Available: https://www.gsma.com/

[2] A. Islam and S. Y. Shin, "A digital twin-based drone-assisted secure data aggregation scheme with federated learning in artificial intelligence of things," *IEEE Netw.*, vol. 37, no. 2, pp. 278–285, Mar. 2023.

[3] P. Mall, R. Amin, A. K. Das, M. T. Leung, and K. R. Choo, "PUF-based authentication and key agreement protocols for IoT, WSNs, and smart grids: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8205–8228, Jun. 2022.

[4] P. Kumar, R. Kumar, G. Srivastava, G. P. Gupta, R. Tripathi, T. R. Gadekallu, and N. N. Xiong, "PPSF: A privacy-preserving and secure framework using blockchain-based machine-learning for IoT-driven smart cities," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2326–2341, Jul. 2021.

[5] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.

[6] X. Xiang, M. Wang, and W. Fan, "A permissioned blockchain-based identity management and user authentication scheme for E-health systems," *IEEE Access*, vol. 8, pp. 171771–171783, 2020.

[7] S. He, Z. Li, J. Wang, and N. N. Xiong, "Intelligent detection for key performance indicators in industrial-based cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5799–5809, Aug. 2021.

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[9] J. B. Xue and Z. M. Bai, "Security and efficient authentication scheme for mobile edge computing," *J. Beijing Univ. Posts Telecommun.*, vol. 44, no. 1, pp. 110–116, Jan. 2021.

[10] O. Salman, S. Abdallah, I. H. Elhajj, A. Chehab, and A. Kayssi, "Identity-based authentication scheme for the Internet of Things," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Messina, Italy, Jun. 2016, pp. 1109–1111.

[11] K. Xue, P. He, X. Zhang, Q. Xia, D. S. L. Wei, H. Yue, and F. Wu, "A secure, efficient, and accountable edge-based access control framework for information centric networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1220–1233, Jun. 2019.

[12] P. Black and R. Layton, "Be careful who you trust: Issues with the public key infrastructure," in *Proc. 5th Cybercrime Trustworthy Comput. Conf.*, Auckland, New Zealand, Nov. 2014, pp. 12–21.

[13] J. Ni, K. Zhang, X. Lin, and X. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2018.

[14] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 858–880, 1st Quart., 2019.

[15] B. Cao, Y. Li, L. Zhang, L. Zhang, S. Mumtaz, Z. Zhou, and M. Peng, "When Internet of Things meets blockchain: Challenges in distributed consensus," *IEEE Netw.*, vol. 33, no. 6, pp. 133–139, Nov. 2019.

[16] S. Matsumoto and R. M. Reischuk, "IKP: Turning a PKI around with decentralized automated incentives," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 410–426.

[17] A. Garba, Q. Hu, Z. Chen, and M. R. Asghar, "BB-PKI: Blockchain-based public key infrastructure certificate management," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun.; IEEE 18th Int. Conf. Smart City; IEEE 6th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Yanuca Island, Fiji, Dec. 2020, pp. 824–829.

[18] A. Garba, Z. Chen, Z. Guan, and G. Srivastava, "LightLedger: A novel blockchain-based domain certificate authentication and validation scheme," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1698–1710, Apr. 2021.

[19] P. Gu and L. Chen, "An efficient blockchain-based cross-domain authentication and secure certificate revocation scheme," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Dec. 2020, pp. 1776–1782.

[20] W. Wang, N. Hu, and X. Liu, "BlockCAM: A blockchain-based cross-domain authentication model," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Guangzhou, China, Jun. 2018, pp. 896–901.

[21] C. Yuan, W. Zhang, and X. Wang, "EIMAKP: Heterogeneous cross-domain authenticated key agreement protocols in the EIM system," *Arabian J. Sci. Eng.*, vol. 42, no. 8, pp. 3275–3287, Aug. 2017.

[22] D. He, S. Chan, and M. Guizani, "An accountable, privacy-preserving, and efficient authentication framework for wireless access networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1605–1614, Mar. 2016.

[23] D. He, J. Bu, S. Chan, C. Chen, and M. Yin, "Privacy-preserving universal authentication protocol for wireless communications," *IEEE Trans. Wireless Commun.*, vol. 10, no. 2, pp. 431–436, Feb. 2011.

[24] L. Wang, Y. Tian, and D. Zhang, "Toward cross-domain dynamic accumulator authentication based on blockchain in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2858–2867, Apr. 2022.

[25] M. Wang, L. Rui, Y. Yang, Z. Gao, and X. Chen, "A blockchain-based multi-CA cross-domain authentication scheme in decentralized autonomous network," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 2664–2676, Sep. 2022.

[26] X. Jia, N. Hu, S. Su, S. Yin, Y. Zhao, X. Cheng, and C. Zhang, "IRBA: An identity-based cross-domain authentication scheme for the Internet of Things," *Electronics*, vol. 9, no. 4, p. 634, Apr. 2020.

[27] S. Guo, F. Wang, N. Zhang, F. Qi, and X. Qiu, "Master-slave chain based trusted cross-domain authentication mechanism in IoT," *J. Netw. Comput. Appl.*, vol. 172, Dec. 2020, Art. no. 102812.

[28] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, "A blockchain-based mutual authentication scheme for collaborative edge computing," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 146–158, Feb. 2022.

[29] S. Showkat Moni and D. Manivannan, "A lightweight privacy-preserving V2I mutual authentication scheme using cuckoo filter in VANETs," in *Proc. IEEE 19th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, Jan. 2022, pp. 815–820.

[30] J. Qi, "Research on the application of accumulator in blockchain," M.S. thesis, Nanjing Univ. Inf. Eng., Nanjing, China, 2020.

[31] M. X. Miao, P. R. Wu, and Y. L. Wang, "Research progress and application of password accumulator," *J. Xidian Univ.*, vol. 49, no. 1, pp. 79–91, Sep. 2022.

[32] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[33] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[34] Z. Liao, X. Pang, J. Zhang, B. Xiong, and J. Wang, "Blockchain on security and forensics management in edge computing for IoT: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 1159–1175, Jun. 2022.

[35] H. K. Jiang et al., "Improved certificateless proxy blind signature scheme with forward security," *Comput. Sci.*, vol. 48, no. 6A, pp. 529–532, Jun. 2021.

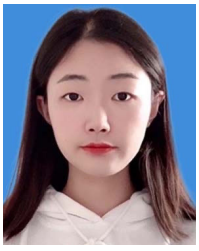
[36] N. Kahya, N. Ghoualmi, and P. Lafourcade, "Formal analysis of PKM using scyther tool," in *Proc. Int. Conf. Inf. Technol. e-Services*, Sousse, Tunisia, Mar. 2012, pp. 1–6.

[37] G. Thakur, P. Kumar, Deepika, S. Jangirala, A. K. Das, and Y. Park, "An effective privacy-preserving blockchain-assisted security protocol for cloud-based digital twin environment," *IEEE Access*, vol. 11, pp. 26877–26892, 2023.

[38] J. Ryu, S. Son, J. Lee, Y. Park, and Y. Park, "Design of secure mutual authentication scheme for metaverse environments using blockchain," *IEEE Access*, vol. 10, pp. 98944–98958, 2022.

[39] M. Bellare, R. Canetti and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols," in *Proc. 30th Annu. ACM Symp. Theory Comput. (STOC)*, May 1998, pp. 419–428.

[40] *FISCO-BCOS*. Accessed: Oct. 2020. [Online]. Available: http://www.fisco-bcos.org



YAO CHEN received the B.S. degree from the School of Electronic and Information Engineering, Weifang Institute of Technology, in 2015. She is currently pursuing the M.S. degree with the School of Mathematics and Computer Science, Dali University. Her research interests include blockchain technology and the IoT security.

QINGQING YANG received the M.S. degree from the School of Mathematics and Computer Science, Dali University, in 2022. Her research interests include blockchain security and cloud computing security.

DENGQI YANG received the B.S. and M.S. degrees in computational mathematics from Yunnan University, Kunming, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer science from Sichuan University, Chengdu, China, in 2012. He is currently a Professor of computer science with the College of Mathematics and Computer, Dali University, Dali, China. His main research interests include digital signatures, identity authentication, and artificial intelligence.

XIN ZENG received the M.S. degree from the School of Information Science, Yunnan University, in 2013. He is currently a Lecturer with the School of Mathematics and Computer Science, Dali University. His main research interests include spatial juxtaposition, pattern mining, and association rule mining.

XIAOWEI LI received the B.S. degree in mathematics and applied mathematics and the Ph.D. degree in information security from Xidian University, China, in 2008 and 2013, respectively. He is currently an Associate Professor with the School of Mathematics and Computer Science, Dali University. His main research interests include cybersecurity protocols, cloud computing security, and blockchain technology.