Journal of Information Security and Applications 56 (2020) 102538

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

BTCAS: A Blockchain-Based Thoroughly Cross-Domain Authentication Scheme

Hongxia Zhang[a], Xingshu Chen[a,*], Xiao Lan[a,*], Hongjian Jin[a], Qi Cao[a]

College Of Cybersecurity, Sichuan University, Chengdu, China

ARTICLE INFO

Article history:

Keywords:
Thoroughly cross-domain
Entity authentication
Blockchain technology
Smart contract
Hyperledger

ABSTRACT

In many real resource access scenarios, the parties who require to establish communication may be unable to effectively identify and verify some authentication messages of the other party due to the use of completely different cryptography settings, making it difficult to authenticate each other in such scenarios. We usually define the above-mentioned problem as "cross-domain authentication". Although many research works have been devoted to solving the problem of entity authentication for cross-domain communication, the problem of "incomplete cross-domain" widely exists in existing works. Existing solutions based on some common underlying cryptography foundations greatly limit the application of such cross-domain authentication schemes. In order to solve the problem of "incomplete cross-domain", this paper proposes a thoroughly cross-domain authentication scheme based on blockchain technology, which can be used by participants from different domains that adopt totally different settings. Our security analysis demonstrates that the scheme is provably secure in the standard model and the experimental results show the efficiency of our scheme.

1. Introduction

1.1. Background

To prevent an impostor from impersonating a legitimate entity to acquire the access privilege, a method for identity authentication is needed in the communication process [1]. In practice, especially in the distributed network environment, to protect the resources and facilitate centralized management of users belonging to the same organization, each organization will form an independent trust domain [2] (e.g., Organization A and Organization B in Figure 1). In the above scenario, every trust domain would have its users and resources (e.g., Server A_1 and Device A_2 for Organization A). However, user or device (e.g., Device A_2) from a specific domain may need a variety of resources that may not be offered by a single domain, one domain has to request another domain or multiple domains. To prevent attacks initiated by unknown identity devices, identity authentication is required in the process of cross-domain communication.

There are many practical scenarios for cross-domain authentication, such as cross-domain authentication over wireless local area networks (WLANs) [3], device authentication in Multi-Domain Home Networks [4] and cross-domain authentication for medical system [5]. To solve the problem of cross-domain authentication, many solutions have been proposed. In summary, these solutions can be divided into three categories: PKI (Public Key Infrastructure)-based scheme, IBE (Identity-Based Encryption)-based scheme, and recently introduced blockchain-based work. Existing PKI-based schemes mainly rely on the certificate authority (CA) to manage digital certificates for users [6], which may lead to a high cost of certificate management, and more, such PKI-based schemes may not scale well [7]. By removing the transmission, verification, and maintenance of the certificate, the IBE-based scheme [8] tends to be a potential solution for cross-domain authentication. However, designing an efficient, secure and lightweight IBE-based scheme remains challenging. By introducing blockchain to cross-domain authentication, the blockchain-based work can achieve the goal of cross-domain authentication more effectively and safely. However, all the schemes mentioned above require users to adopt a common cryptographic settings such as the same hash algorithm or signature algorithm, which can be a barrier when users coming from different cryptographic settings wish to communicate with each other [9].

In this paper, we suggest the "thoroughly cross-domain" referring to the cross-domain scheme allows different domains to adopt various cryptographic settings such as different hash algorithms and signature algorithms, and the "incompletely cross-domain" refers to opposite meaning. Aiming at tackling the challenge of "incompletely cross-domain", we propose the first

* Corresponding authors.

https://doi.org/10.1016/j.jisa.2020.102538
2214-2126/

blockchain-based thoroughly cross-domain authentication scheme (we call it BTCAS) which allows users to come from different cryptographic settings without relying on Trusted Third Party (TTP).

1.2. Technical contributions

To better counteract the problem of "incompletely cross-domain", inspired by [9], the first blockchain-based thoroughly cross-domain authentication scheme (BTCAS) is proposed in this paper. In BTCAS, participants from different domains (with various cryptographic settings) communicating over an insecure channel can establish a trust relationship with each other. Our detailed contributions are as follows:

1) We propose the first blockchain-based thoroughly cross-domain authentication scheme (BTCAS), which can perfectly solve the challenge of "incompletely cross-domain". In general, a TTP needs to be introduced because cryptographic settings adopted by participants may be different. While in BTCAS, the participants coming from different domains can directly call the chaincode which has been installed in the blockchain network to independently perform the calculation such as digital signature or hash. In this way, the reliability of the returned result can be ensured while getting rid of the introduction of TTP.
2) Our scheme can massively reduce the computation burden of the authentication server since all the verification calculations of our scheme are performed by chaincode. In the process of cross-domain authentication, the authentication server just needs to send some query messages to chaincode to get the authentication result.
3) We formally analyzed the proposed BTCAS and it is provably secure in the standard model.

1.3. Outline

The rest of this paper is organized as follows. In Section 2, we revisit the related works to cross-domain authentication. Section 3 presents the related theoretical knowledge, such as blockchain technology, signature scheme, and hash function. The system model and proposed scheme are shown in Section 4. Section 5 and Section 6 present the security and performance of BTCAS respectively. Finally, we draw our conclusions in Section 7.

2. Related work

Over the past several years, there has been a lot of works on cross-domain authentication. There are two strands of research related to this paper.

Traditional solutions. Based on traditional PKI, in order to achieve the goal of cross-domain authentication, a two-layered PKI model was proposed in [4] which are the Global PKI layer and Local PKI layer to realize a scalable, efficient registration and authentication. However, as a typical hierarchical structure, the breakdown of the Global PKI layer will incur disastrous destruction [10]. A bridge CA cross-certification model (BCA) was introduced and analyzed in [11] where the trustworthy independent node BCA can establish trust relationships with other non-related CAs. This BCA-based model can reduce the number of possible certificate paths, but the breakdown of BCA is also a fatal to the whole system. Later, Zhang et al. [12] presented a novel distributed trust model based on a virtual BCA which has the same advantages as the traditional BCA model but requires no cost of the physical BCA. However, all of the PKI-based schemes [4,11,12] necessitate the need for certificate authority (CA) to store, issue, and manage the digital certificates for each user [6].

Based on identity-based encryption (IBE), McCullagh et al. [13] proposed an efficient cross-domain two-party construction. In 2010, Cao et al. [14] introduced a more efficient protocol that minimizes message exchange time with no extra cost. Both works [13,14] can handle the problem of certificate management and transmission overhead existing in traditional PKI-based schemes. However, the requirement of the TTP (e.g., PKG) and the problem of "incompletely cross-domain" may be a bottleneck when applied in practical scenarios. Recently, Yuan et al. [15] realized a key agreement protocol between the PKI domain and the IBC (identity-based cryptography) domain, but their protocol still requires that all the participants use parameters from the same cryptographic settings, which is still "incompletely cross-domain".

Based on indistinguishability obfuscation, in 2016, Lan et al. [9] proposed a one round cross-domain group key exchange protocol which solved the problem of "incompletely cross-domain". The basic idea of this protocol is the following: 1) Setup phase: a TTP chooses a key for a constrained PRF (pseudorandom function) and publishes a program indistinguishability obfuscator for PRF as the global agreed domain parameter; 2) Group key change phase: each party P generate a signature of a random x using his own signature scheme, then broadcast the signature. 3) Group key generation phase: By running PRF on the input of concatenation of identity P and x, the session key obtained. However, a TTP is also needed in their protocol in the setup phase.

Blockchain based solutions. The emergence of blockchain technology[16] and Smart Contract[17] further simplifies the cross-domain authentication process. In [18], by storing the hash value of domain root CA certificate into the blockchain, Zhou et al. proposed a cross-domain authentication scheme compatible with the current PKI system, which not only improves the efficiency of the authentication process but also enhances the scalability. A similar authentication scheme was proposed in [19], which added the digital signature algorithm based on SM9 into the authentication process to further improve the security. In the resource sharing scenarios between different companies or institutions, Wang et al. [20] proposed a cross-domain framework based on consortium blockchain technology, its basic idea is also similar to [18]. By introducing blockchain to the environment of IoT and cloud, secure authentication of IoT devices and cloud identity management are implemented respectively in [21] and [22]. To solve the problem of cross-data center authentication for distributed VFS, Yao et al. [23] proposed a blockchain-assisted lightweight anonymous authentication mechanism. Subsequently, an anonymous cross-domain authentication scheme for the medical PKI system was designed in [5] and a novel blockchain-based cross-domain authentication scheme for Wi-Fi access was introduced in [24]. By replacing the TTP with blockchain and storing the data which needs to retain integrity to the blockchain, the schemes [5,18–24] are safer and more efficient than the traditional solutions mentioned before. But all of the schemes do not take advantage of the Smart Contract which can operate safely and independently of third-party control, and above all, they all suffer from a common problem of "incomplete cross-domain" that all participants are required to adopt the same cryptographic setting.

3. Preliminaries

In this section, we mainly introduce the basic theoretical knowledge involved in our paper. We start by providing a brief introduction to the blockchain technology and explain why the consortium blockchain is chosen as the basis of our scheme instead of the private chain or public chain. Then we briefly recall the definitions of the cryptographic primitives essential for our study.

Table 1
Comparison of Technical Characteristics of Various Blockchains

Blockchain	Degree of Centralization	Consensus Area	Consensus Mechanism	Cost	Efficiency	Scalability
Public Chain	Centerless	Any Person or Institution	POW/POS	High	Low	Low
Consortium Chain	Multi-Center	Inter Institution	PBFT	Medium	Medium	Medium
Private Chain	Single Center	Within Institution	RPCA	Low	High	High

3.1. Blockchain technology

Blockchain technology was first introduced in [25] as the underlying technology of Bitcoin. Afterward, the advent of other cryptocurrencies, like Ethereum [26] and Zcash [27], accelerated the development of blockchain technology. A blockchain is fundamentally a distributed, shared, and immutable database ledger that stores transactions across a peer-to-peer (P2P) network [28]. Blockchain technology combines a series of computer and cryptography technologies, such as distributed storage, peer-to-peer communication, consensus mechanisms, and asymmetric cryptographic algorithms, to achieve a highly trusted network that does not require the participation of TTP [29]. As shown in Table 1, the blockchain systems can be categorized roughly into three types [16] such as Public Blockchain (e.g., Ethereum [26]), Private Blockchain and Consortium Blockchain (e.g., Hyperledger [30]) according to whether the network has access permission mechanism and whether the subject with control rights is concentrated.

Currently, the research works based on blockchain technology mainly utilize the following three characteristics:

1) The data stored on the blockchain are immutable and thus trustworthy;
2) The transaction on the blockchain can be retrieved;
3) Some blockchain platforms (e.g., Ethereum [31]) implement the Smart Contract [17], which is a computerized transaction protocol that executes in terms of the contract. Because smart contracts allow us to have general-purpose computations occur on the chain [32], users can do more with the blockchain. Although the Smart Contract is not exempt from all risks [33], we usually assume that it is sufficiently secure for executing the agreed clauses.

Considering the functional requirements of our cross-domain authentication scheme: 1) The scope of consensus is across organizations; 2) The efficiency and scalability of blockchain should be acceptable; 3) The Smart Contract involved in the blockchain should be functional so as to support the calculation of basic cryptographic algorithms like digital signature and hash function. In summary, we chose to design the BTCAS based on the consortium blockchain.

As a typical consortium blockchain - HyperLedger [30] is a distributed ledger technology framework dedicated to the development of cross-industry commercial blockchain platform technology. Compared with the public chain like Bitcoin or Ethereum, HyperLedger integrates the Membership Service Provider (MSP) and has better performance [34]. It's worth noting that the Smart Contract involved in HyperLedger is implemented by Go language [35], which means the developers can implement any logic implemented by Go language under the top of HyperLedger. And the Hyperledger allows the client to subscribe to custom events produced by a chaincode, the client will catch the event once the ledger is updated.

3.2. Signature scheme

A generic digital signature scheme is a tuple of probabilistic polynomial-time algorithms $SIG = (Gen, Sign, Verify)$ [36]. The key generation algorithm $(pk, sk) \leftarrow Gen(1^\lambda)$ outputs a public key pk used to verify the signature and a secret key sk used to sign the signature on input of security parameter λ. The signing algorithm $\sigma \leftarrow Sign(sk; m)$ generates a signature σ responds to a message m from specific message space with the input of sk. Given a signature σ, a message m and a public key pk, the verification algorithm $Verify(pk; m; \sigma)$ returns 1 if σ is a valid signature of m, and 0 otherwise.

Correctness. For every λ, every $(pk, sk) \leftarrow Gen(1^\lambda)$, and every message m from specific message space, we require that $Verify(pk;m,Sign(sk;m))=1$.

Security. Consider the following experiment $\mathrm{Expt}^{forge}_{\mathcal{A},SIG}(1^\lambda)$ played between a challenger $\mathcal{A}$ and an adversary $\mathcal{A}$.

1. Challenger $\mathcal{C}$ obtains (pk,sk) by running $Gen(1^\lambda)$.
2. Adversary $\mathcal{A}$ is given pk, and can query arbitrary m to challenger $\mathcal{C}$. Challenger $\mathcal{C}$ returns a signature $\sigma = Sign(sk;m)$.
3. Adversary $\mathcal{A}$ then outputs $(m;\sigma)$, the output of the experiment is 1 if $m \notin \mathcal{Q}$($\mathcal{Q}$ denote the messages queried by adversary $\mathcal{A}$ during the execution) and $Verify(pk;m;\sigma) = 1$.

Definition 1. We say that the signature scheme SIG=(Gen,$Sign$,$Verify$) is *existentially unforgeable under an adaptive chosen-message attack* if for any polynomial-time adversary $\mathcal{A}$, there exists a negligible function $negl$ such that:

$Pr[\mathrm{Expt}^{forge}_{\mathcal{A},SIG}(1^\lambda) = 1] \leq negl(\lambda)$

3.3. Hash function

A general hash function which compresses an arbitrary-length strings to fixed-length strings includes a pair of polynomial-time algorithms (Gen, H) [37]. Algorithm Gen outputs a key $s(\lambda)$ with the input of security parameter 1^λ. And there exists a polynomial ℓ such that polynomial-time algorithm H outputs $H^s(x) \in \{0, 1\}^{\ell(\lambda)}$ on input of any string $x \in \{0, 1\}^*$ and a key $s(\lambda)$.

Security. The security is defined against the following experiment $\mathrm{Expt}^{secondpre}_{\mathcal{A},\Pi}(1^\lambda)$ played between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. The challenger $\mathcal{C}$ generates a key $s(\lambda) \leftarrow Gen(1^\lambda)$ and a value x, the adversary $\mathcal{A}$ receives $s(\lambda)$ and x as input.
2. The adversary $\mathcal{A}$ outputs x' ($x' \neq x$).
3. The output of the experiment is 1 if and only if $H^s(x) = H^s(x')$.

Definition 2. We say that a hash function $HASH = (Gen, H)$ is *second preimage resistant* if for any probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function $negl$ such that

$Pr[\mathrm{Expt}^{secondpre}_{\mathcal{A},\Pi}(1^\lambda) = 1] \leq negl(\lambda)$

Definition 3(**Preimage resistance** [37]). The hash function $HASH$ = (Gen,H) is *preimage resistant* if the following two conditions hold:

1. Easy to compute: There exists a polynomial-time algorithm M_H such that on input any $x \in \{0, 1\}^*$, M_H outputs $H^s(x)$.
2. Hard to invert: For any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function $negl$ such that
   $P_r[\mathcal{A}(H^s(x)) \in H^{-s}(H^s(x))] \leq negl(\lambda)$
   where the probability is taken over the uniform choice of x in $\{0, 1\}^\lambda$, any $s(\lambda) \leftarrow Gen(1^\lambda)$ and the random coin tosses of $\mathcal{A}$.

Table 2
The description of protocol symbol

Symbol	*Symbol descriptions*
U_{X_i}	The i-th User of Domain X
AS_X, AS_Y	Authentication server in domain X and Y
CC	The chaincode deployed on the blockchain
pk_Z, sk_Z	Public key and private key of Entity Z
$Sign_X$	Signature algorithm used in Domain X
$Hash_X$	Hash algorithm used in Domain X
$\|\|$	Message concatenation operation
$(m_1\|m_2)$	The key-value pair format where m_1 is the key and m_2 is the value

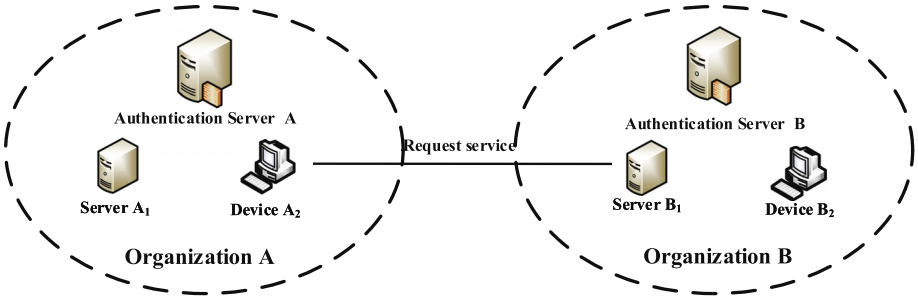


Fig. 1. The Scene of Cross-Domain Communication.

4. Design of BTCAS

At the very beginning, we give the symbols and their corresponding meanings which will be involved in the following work as shown in Table 2. In this section, we first present the system model (in Section 4.1), then we will introduce our Blockchain-Based Thoroughly Cross-Domain Authentication Scheme (BTCAS) in detail, which involves initialization of the system (in Section 4.2), intra-domain authentication (in Section 4.3) and the most important part: cross-domain authentication (in Section 4.4). To explain briefly and accurately, we take the cross-domain authentication process between users from domain A and domain B as an example.

4.1. System model

In order to achieve the goal of thoroughly cross-domain authentication, we design a thoroughly cross-domain authentication system model based on blockchain as shown in Figure 2, which consists of domains, authentication servers, users, and a blockchain network.

Domain : We define a domain as a trust scope that users in the same domain trust each other. As shown in Figure 2, the domains are represented by Domain A and B. The trust scopes of different domains are independent, and we also indicate that different domains may adopt different cryptographic settings.

Authentication Server : The authentication server mainly processes authentication requests from users in the same (Intra-domain authentication) or different (Cross-domain authentication) domains, each domain has only one authentication server. As shown in Figure 2, the authentication servers in domains A and B are represented by AS_A and AS_B respectively.

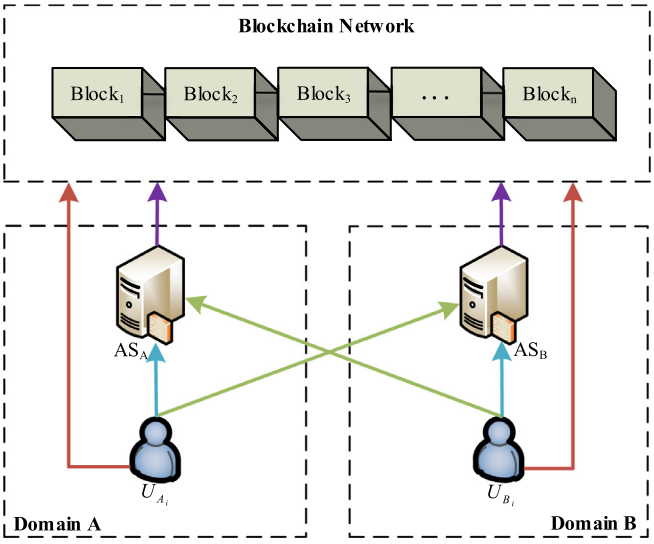


Fig. 2. Thoroughly cross-domain authentication system model

User : The user can be described as entity that owns specific resources within a domain and may have need to access other users in other domains. As shown in Figure 2, the users in domains A and B are represented by U_{A_i} and U_{B_i}.

Blockchain Network : As shown in Figure 2, after being permitted, the authentication servers and users from different domains join the blockchain network and can interact with blockchain network. Our cross-domain authentication system model is built on the top of the consortium chain Hyperledger.

4.2. Initialization

In the initialization phase of our system, the public and private keys for all entities in domains A and B are generated, the blockchain network is established, the chaincode is deployed and the cryptographic settings information adopted by a certain domain is stored in the blockchain network. This phase is executed only once.

Step I1: The establishment of the public and private key.

All entities in domain A and domain B, including the authentication servers and all users, initialize their public and private keys according to the cryptographic setting adopted in that domain.

Step I2: The establishment of blockchain network and deployment of chaincode.

After being permitted, AS_A and AS_B join the blockchain network, and the chaincode *CC* will be deployed to the blockchain. The deployed chaincode *CC* designed by us includes three functions, each function can be triggered with its function identifier (its name) and the necessary parameters:

• *PublishDomain*($Hash_X(pk_{AS_X})$, $Sign_X$, $Hash_X$): this function handles domain cryptographic settings information storage requests coming from authentication servers which joined the blockchain network beforehand. It can be triggered with a function identifier *PublishDomain* and the enumerated parameters, where $Hash_X(pk_{AS_X})$ is the hash value of the pk_{AS_X}, $Sign_X$ and $Hash_X$ represent the signature and hash algorithm adopted in domain *X* respectively.

• *PublishUser*($Hash_X(pk_{U_{X_i}})$, $Hash_X(pk_{AS_X})$, *state*): this function handles user identity information storage requests coming from authentication servers which joined the blockchain network beforehand. It can be triggered with a function identifier *PublishUser* and the enumerated parameters, where $Hash_X(pk_{U_{X_i}})$ and $Hash_X(pk_{AS_X})$ represent the hash value of $pk_{U_{X_i}}$ and pk_{AS_X} which calculated by hash algorithm adopted in domain *X*, *state* indicates whether the user identity information is available.

• *Verify*($pk_{U_{X_i}}$, $Sign_X(sk_{U_{X_i}}; N)$, *N*, $Hash_X(pk_{AS_X})$): this function handles user identity verify request coming from authentication servers which joined the blockchain network beforehand. It can be triggered with a function identifier *Verify* and the enumerated parameters, where $pk_{U_{X_i}}$ represents the public key of U_{X_i}, $Sign_X(sk_{U_{X_i}}; N)$ is the signature of *N* with the signature algorithm $Sign_X$, *N* is a random number and $Hash_X(pk_{AS_X})$ indicates the hash value of pk_{AS_X}.

Step I3: Store the domain cryptographic settings information to blockchain.

In this step, we store the domain cryptographic settings information ($Hash_X(pk_{AS_X})$, $Sign_X$, $Hash_X$) to blockchain, here are the details:

AS $_X$: After joined the blockchain network, in order to store the domain cryptographic settings information of domain *X* to blockchain network, AS_X sends a request (*PublishDomain*, ($Hash_X(pk_{AS_X})$, $Sign_X$, $Hash_X$)) to *CC*.

CC: Upon receiving the request (*PublishDomain*, ($Hash_X(pk_{AS_X})$, $Sign_X$, $Hash_X$)) from AS_X, *CC* executes function *PublishDomain*($Hash_X(pk_{AS_X})$, $Sign_X$, $Hash_X$) to process the request. Inside this function, *CC* stores the tuple to blockchain network as the form of key-value pair ($Hash_X(pk_{AS_X})$| $Sign_X$, $Hash_X$).

4.3. Intra-domain authentication

To manage the user identity within a domain in a more granular manner, similar to the process of the traditional CA, the operations of users' identity in our scheme also involve the process of *Application, Verification, Update*, and *Revocation*.

Application. In the phase of application, all users U_{X_i} in domain A and domain B send an identity application to AS_X of that domain for identity register. The format of identity application is ($pk_{U_{X_i}}$, *application,* σ), where $pk_{U_{X_i}}$ is the public key of U_{X_i}, status = "*application*" means this message is a request for authentication, $\sigma = Sign_X(sk_{U_{X_i}}; Hash_X(pk_{U_{X_i}}))$ is the signature of $Hash_X(pk_{U_{X_i}})$ signed by U_{X_i}.

Verification. Upon receiving and verifying the message from U_{X_i}, AS_X will store or update the user's identity information to the blockchain, and register or revoke the identity in the blockchain network for users as follows:

(1) AS_X authenticates the message:

• Check whether the message is in the correct format.

• Verify the correctness of the user's signature.

(2) If any of the above verifications fails, AS_X reports failure to U_{X_i} and aborts. Otherwise, AS_X will check the status:

• *Status* = *"application"*: After checking that the $pk_{U_{X_i}}$ has never been registered before (otherwise reports failure and aborts), AS_X sets $pk = pk_{U_{X_i}}$, *state* ="*true*", and registers a new identity in the blockchain network for U_{X_i}.

• *Status* = *"update"*: After checking that the $pk^{new}_{U_{X_i}}$ has never been registered before(otherwise reports failure and aborts), AS_X sets $pk = pk^{new}_{U_{X_i}}$, *state* ="*true*".

• *Status* = *"revocation": ASX* sets $pk = pk_{U_{X_i}}$, *state* ="*false*", and revokes the identity in the blockchain network for U_{X_i}.

(3) AS_X stores the identity information of U_{X_i} to blockchain:

• Compute h=$Hash_X(pk)$ using the hash algorithm $Hash_X$ adopted in domain *X*.

• AS_X initiates a blockchain transaction to send user identity information storage request (*PublishUser*,(h, $Hash_X(pk_{AS_X})$, *state*)) to *CC*. Upon receiving the request from AS_X, *CC* executes function *PublishUser*(h, $Hash_X(pk_{AS_X})$, *state*) to store user identity information to blockchain as the key-value pair format (h| $Hash_X(pk_{AS_X})$, *state*).

Update. When updating the identity information out of some requirements, U_{X_i} needs to submit an identity update request($pk^{old}_{U_{X_i}}$, $pk^{new}_{U_{X_i}}$, *update,* σ_1, σ_2) to AS_X, where $pk^{old}_{U_{X_i}}$ and $pk^{new}_{U_{X_i}}$ indicate the old and new public key of U_{X_i}, *status* =" *update*" means that this is an update request, σ_1=$Sign_X(sk^{old}_{U_{X_i}}; Hash_X(pk^{old}_{U_{X_i}} || pk^{new}_{U_{X_i}}))$, σ_2=$Sign_X(sk^{new}_{U_{X_i}}; Hash_X(pk^{old}_{U_{X_i}}))$.

Revocation. Before the leaving of U_{X_i} from domain *X*, the identity revocation request must be submitted to the AS_X to revoke the identity information which stored in blockchain, the user's identity in the blockchain network also need to be revoked. The format of identity revocation request is ($pk_{U_{X_i}}$, *revocation,* σ), where $pk_{U_{X_i}}$ is the public key of U_{X_i}, status = "*revocation*" means that this message is a request for revocation, $\sigma = Sign_X(sk_{U_{X_i}}; Hash_X(pk_{U_{X_i}}))$ is the signature of $Hash_X(pk_{U_{X_i}})$ signed by U_{X_i}.

4.4. Cross-domain authentication

As shown in Figure 3, our blockchain-based cross-domain authentication protocol comprises six steps which are Step C1-C6 below and among these steps, Step C5 (Verification) executed by chaincode *CC* is the most important and intricate part of our protocol. During the whole cross-domain authentication process, we should note that AS_B just needs to do simple interactions with U_{A_i} and *CC* to authenticate U_{A_i}, all the complicated calculations will be executed by *CC* which is the reason why our BTCAS can reduce the burden of the authentication server. The specific authentication process between U_{A_i} and AS_B is as follows.

Step C1: U_{A_i} requests AS_B to authenticate.

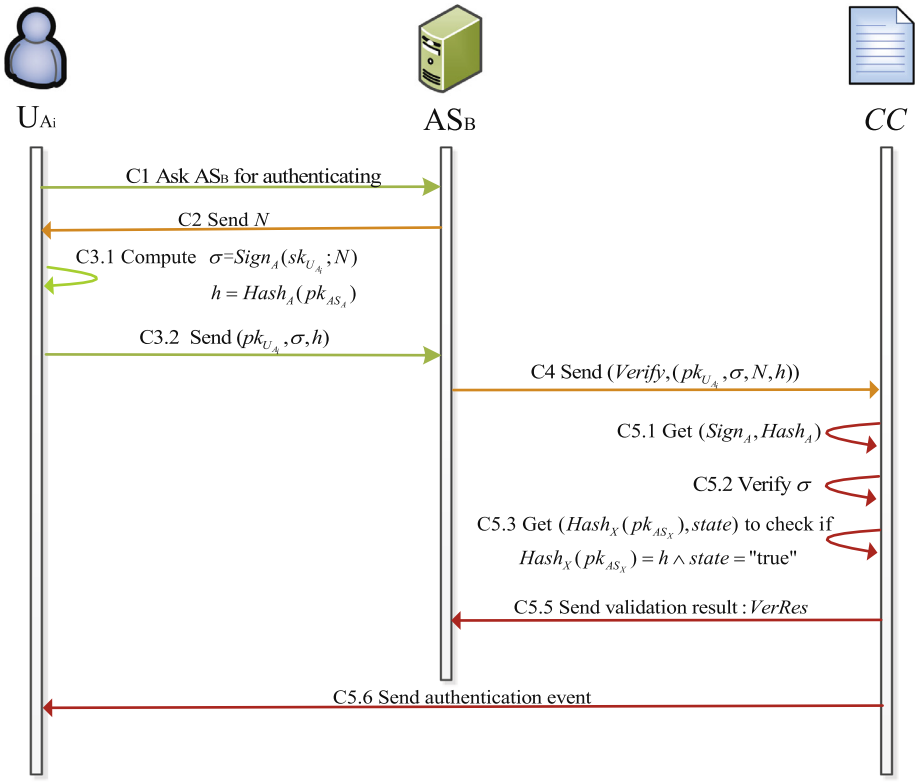
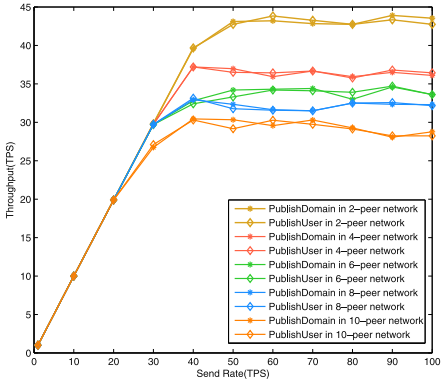
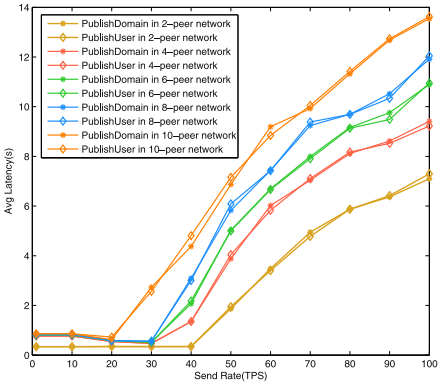


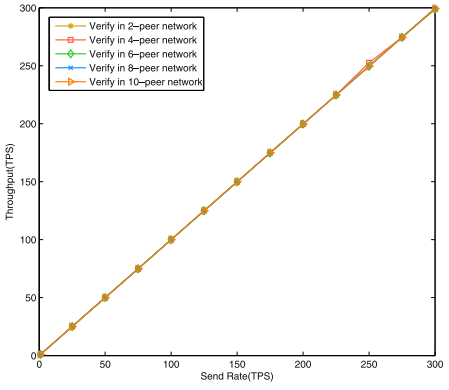
Fig. 3. Cross-domain authentication protocol



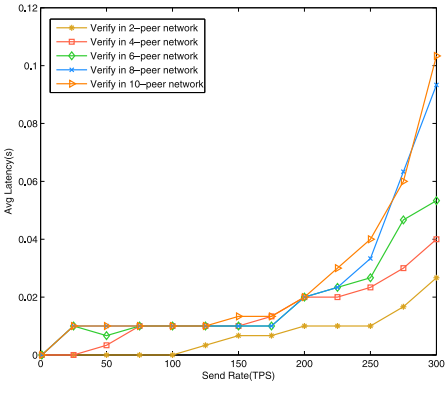
(a) Throughput of Invoke operation



(b) Avg latency of Invoke operation



(c) Throughput of Query operation



(d) Avg latency of Query operation

Fig. 4. Caliper throughput and average transaction latency of BTCAS in different network environments

Step C2: Upon receiving authenticate request from U_{A_i}, AS_B responds to U_{A_i} with a random number N which obtained by running RandomGen(1^λ).

Step C3: Upon receiving N from AS_B, U_{A_i} performs the following:

(1) U_{A_i} computes

$\sigma = Sign_A(sk_{U_{A_i}}; N)$.

$h = Hash_A(pk_{AS_A})$.

(2) U_{A_i} sends ($pk_{U_{A_i}}$, σ, h) to AS_B.

Step C4: In this step, AS_B initiates a blockchain transaction to send the following messages: the function identifier *Verify* which is waiting to be executed, a parameter tuple ($pk_{U_{A_i}}$, σ, N, h) which is necessary for *CC* to run its function *Verify*.

Step C5: Upon receiving the tuple (*Verify*,($pk_{U_{A_i}}$, σ, N, h)), *CC* executes function *Verify*($pk_{U_{A_i}}$, σ, N, h) to obtain the verify result-*VerRes* and sets event to U_{A_i}. The detail are as follows:

(1) Get the domain cryptographic settings information:

At the phase of system initialization (Section 4.2), the information of digital signature algorithm $Sign_A$ and hash algorithm $Hash_A$ adopted in domain A have been saved to blockchain as a form of key-value pair ($Hash_A(pk_{AS_A})|Sign_A$, $Hash_A$), so as the value ($Sign_A$, $Hash_A$) can be directly returned by *CC* with the input of h.

• The value returned by *CC* is null. The function *Verify* returns *VerRes* with "Authentication server does not exist" to AS_B and sets event to U_{A_i}, the event parameters include the N and *VerRes*, then aborts *CC*.

• The value returned by *CC* is not null. *CC* continues to perform the next step.

(2) Verify the signature:

After obtaining the value $Sign_A$, the next step is to verify the signature σ, that is, to verify the user interacting with AS_B is indeed the owner of $pk_{U_{A_i}}$. The function *Verify* performs signature verification with the input (N, $pk_{U_{A_i}}$, σ, $Sign_A$).

• The signature verification failed, the function *Verify* returns *VerRes* with "Signature is not true" to AS_B and sets event to U_{A_i}, the event parameters include the N and *VerRes*, then aborts *CC*.

• The signature verification succeeded, *CC* continues to perform the next step.

(3) Verify the user identity information:

This step is to confirm the authenticity of the domain that U_{A_i} claims to belong to, that is, to confirm whether the user corresponding to $pk_{U_{A_i}}$ actually belongs to domain A. At the phase of intra-domain authentication (Section 4.3), the user identity information of U_{A_i} has been saved to blockchain as the form of key-value pair ($Hash_A(pk_{U_{A_i}})$ | $Hash_A(pk_{AS_A})$, *state*). With the inputs of $pk_{U_{A_i}}$ and $Hash_A$ which obtained from (1), $Hash_A(pk_{U_{A_i}})$ can be calculated and used to query corresponding user identity information under normal conditions.

• The result returned by *CC* is null, which means U_{A_i} doesn't belong to any domain in blockchain. The function *Verify* returns *VerRes* with "Domain information is not true: the user in conversation N does not exist" to AS_B and sets event to U_{A_i}, the event parameter is the *VerRes*, then aborts *CC*. Otherwise, continues to perform the next step.

• Checks whether $Hash_X(pk_{AS_X})$ (returned by *CC*) $= h$ (sent from AS_B). If it does not hold, function *Verify* returns *VerRes* = "Domain information is not true: the domain user in conversation N claimed to belong to is not true" back to AS_B and and sets event to U_{A_i}, the event parameter is the *VerRes*, then aborts *CC*. Otherwise, performs the next step.

• Checks the value of *state*. if *state*="false" which means the user identity information is unavailable, function *Verify* returns *VerRes*="Domain information is not true : user in conversation N is unavailable" to AS_B and sets event to U_{A_i}, the event parameter is the *VerRes*, then aborts *CC*. otherwise sets *VerRes*="The information of user in conversation N is true " and proceeds to the next step.

• Sets event to the U_{A_i} which parameter is the *VerRes*, and returns *VerRes* to AS_B.

AS_B: Upon receiving *VerRes* from *CC*, AS_B checks whether this is an error message. If is, AS_B aborts the process of authentication. Otherwise, AS_B authenticates U_{A_i} to access domain B.

U_{A_i}: Upon catching the event from *CC*, U_{A_i} checks whether this event is related to this authentication process by comparing the random number (included in the event) and the random number N (received from AS_B in **Step C2**). If they are equal, U_{A_i} will check whether the *VerRes* is an error message. If is, U_{A_i} aborts the process of authentication and start a new authentication process. Otherwise, U_{A_i} will start to access AS_B.

Step C6: Mutual authentication.

So far, we have completed the unilateral authentication process between U_{A_i} from domain A and AS_B in domain B, now U_{A_i} can access the resource of domain B. However, in some cases, we need to consider one-to-one bi-directional authentication between users such as U_{A_i} and U_{B_i}, which is exactly what **Step C6** is going to do.

To be authenticated by domain A, U_{B_i} in domain B implements the authentication process with AS_A in domain A similar to **Step C1** to **Step C5** that described above. By this way, the trust chains $U_{B_i} \rightarrow AS_A \rightarrow U_{A_i}$ and $U_{A_i} \rightarrow AS_B \rightarrow U_{B_i}$ can be constructed, thus we can get a mutual trust relationship between U_{A_i} and U_{B_i}.

5. Security analysis

Based on the definition and notion of authentication security in [38], the security of BTCAS is proved in this section.

5.1. Security model

The three-party BTCAS protocol is carried out by a set of *parties* $U \in \mathbb{P} = (\mathbb{I} \cup \mathbb{R} \cup \mathbb{S})$, which is initiator (user to be authenticated), responder (authentication server) and server (*CC*), respectively. We assume that $U \in \mathbb{I}$ possess a long-term asymmetric key-pair (sk_U, pk_U). Each party $U \in \mathbb{P}$ can take part in multiple executions of the protocol, both concurrently and subsequently, called a session. We use π_U^i to refer to the ith session at user U. Associated to each session π_U^i, there is a collection of variables that reflects the local state of π_U^i during the protocol.

• sk_U, pk_U: the long-term private/public key of U.

• peers: the list of identities of the intended communication peers of π_U^i.

• α: the accept state of π_U^i, where $\alpha \in$ {running, accepted, rejected}.

• *identifier*: the session identifier included in π_U^i.

During the execution of protocol, a polynomial-time adversary $\mathcal{A}$ can interact with the participants of protocol via the oracle queries, which model on the abilities of an adversary in the real environment. The possible oracle queries are listed in the following:

• Send(π_U^i, m): This query is used to simulate active attacks, in which an adversary may tamper with the message being sent over the public channel. The adversary gets the response message that the protocol would generate upon receipt of message m.

• Execute(U, V, W): This query models the passive attacks in which the adversary eavesdrops the message exchanged among π_U^i, π_V^j and π_W^k in an honest execution of the protocol.

• Corrupt(U): This query returns the long-term secret key of party U which means party U corrupted.

Definition 4 (Transcripts). A transcript T is defined to be a sequence of communication records, where a communication record is a combination of query Send and query Execute made by the

adversary $\mathcal{A}$, together with their responses. Let $\mathbb{T}$ denote the set of all possible transcripts generated by $\mathcal{A}$ running the protocol.

Definition 5 (**Partner function** [38]). A symmetric and monotonic partner function is a polynomial-time function f: $\mathbb{T} \times (\mathbb{P}\backslash\mathbb{S}) \times \mathbb{N} \to ((\mathbb{P}\backslash\mathbb{S}) \times \mathbb{N}) \cup \{\perp\}$ subject to the following requirements:

1.$f(T,U,i) = (V,j) \Rightarrow f(T,V,j) = (U,i)$.

2.$f(T,U,i) = (V,j) \Rightarrow f(T',U,i) = (V,j)$ for all $T \subseteq T'$.

Instead of $f(T,U,i) = (V,j)$, we also write $f_T(\pi_U^i) = (\pi_V^j)$. A session π' is a partner to π if and only if $\pi' \in f_T(\pi)$.

Definition 6 (Soundness of Partner Function). A partner function f is sound if the following holds for all transcript T. If sessions $f'_T(\pi_U^i) = (\pi_V^j)$

- $\pi_U^i.\alpha = \pi_V^j.\alpha =$accepted $\Rightarrow \pi_U^i.identifier = \pi_V^j.identifier, identifier \neq \perp$.
- π_U^i.peers=(V,W), π_V^j.peers=(U,W), and $W \in S$.
- $U \in \mathbb{I} \wedge \mathbb{V} \in \mathbb{R}$ or $U \in \mathbb{R} \wedge \mathbb{V} \in \mathbb{I}$.

Definition 7 (Two-Party Unilateral Entity Authentication (TP-U-EA)). Let Π be a two-party unilateral authentication protocol, and $\mathcal{A}$ be a polynomial-time adversary which interacts with the real protocol Π as defined above. We say that $\mathcal{A}$ violates the security of the Π if one of the following conditions holds:

- Let the role of U be the initiator, there exists a session π_U^i with $\pi_U^i.\alpha =$ accepted, and π_U^i.peers $= V$, but there doesn't exists session π_V^j satisfies $f(\pi_U^i) = \pi_V^j$ before query Corrupt(V).
- Let the role of U be the server, there exists a session π_U^i with $\pi_U^i.\alpha =$ accepted, and π_U^i.peers $= V$, but there doesn't exists session π_V^j satisfies $f(\pi_U^i) = \pi_V^j$.

We denote the advantage of $\mathcal{A}$ in violating the security of Π as $Adv_{\Pi}^{TP-U-EA}(\mathcal{A})$, and we say a two-party unilateral authentication protocol is a TP-U-EA secure protocol if for any polynomial-time adversary $\mathcal{A}$, the $Adv_{\Pi}^{TP-U-EA}(\mathcal{A})$ is negligible.

Definition 8. Let the protocol executed between initiator and server be Π_1 and the protocol executed between server and responder be Π_2. The three-party protocol Π (described in Section 4) is a secure unilateral authentication protocol if, for any polynomial-time adversary $\mathcal{A}$:

(1) Π_1 is a TP-U-EA secure protocol.
(2) Π_2 is a TP-U-EA secure protocol.
(3) Supporting $\pi_{\mathbb{I}}^i$ is an initiator session having intended peers R (responder) and S (server), then: $\pi_{\mathbb{I}}^i.identifier = \pi_{\mathbb{S}}^j.identifier$.

Security. Under the premise that the identifier is contained in the sessions, for a polynomial-time adversary $\mathcal{A}$, let $Succ_{\Pi_1}^{\mathcal{A}}$ be the event that $\mathcal{A}$ passes the authentication of protocol Π_1 successfully, let $Succ_{\Pi_2}^{\mathcal{A}}$ be the event that $\mathcal{A}$ passes the authentication of protocol Π_2 successfully, and let $Succ_{\Pi}^{\mathcal{A}}$ be the event that $\mathcal{A}$ passes the authentication of protocol Π successfully. We can get the relationship: $Pr[Succ_{\Pi}^{\mathcal{A}}] \leq Pr[Succ_{\Pi_1}^{\mathcal{A}}] + Pr[Succ_{\Pi_2}^{\mathcal{A}}]$. The advantage of $\mathcal{A}$ in violating the security of protocol Π is defined as:

$Adv_{\Pi}(\mathcal{A}) = |2Pr[Succ_{\Pi}^{\mathcal{A}}] - 1|$.

5.2. Security proofs

Lemma 1. Suppose the hash function is second preimage resistant, the digital signature algorithm is existentially unforgeable under an adaptive chosen-message attack and the blockchain is secure. Then, the protocol Π_1 executed between initiator and server is a TP-U-EA secure protocol.

Proof. Let $\mathcal{A}$ be a polynomial-time adversary attacking Π_1. We use a hybrid argument to bound the advantage of $\mathcal{A}$. The ***Game*$_0$**, ***Game*$_1$**, ... is defined and the advantage of $\mathcal{A}$ in ***Game*$_i$** is:

$Adv_i(\mathcal{A}) = |2Pr[\mathcal{A}$ succeeds in **Game$_i$**$] - 1|$.

***Game*$_0$**: This is the original game in which $\mathcal{A}$ interacts with the real protocol as defined in Section 5.1.

***Game*$_1$**: This game proceeds as the previous one but aborts if the adversary $\mathcal{A}$ can not make the event **Collide** happen. Therefore,

$|Adv_1(\mathcal{A}) - |Adv_0(\mathcal{A})| < negl(\lambda)$.

Collide: let **Collide** be the event that, $\mathcal{A}$ find a new, valid public key that has the same hash value as the public key of user U.

Proof. By querying Execute, the adversary $\mathcal{A}$ will get the information of protocol Π_1, which including the public key of user U. Assuming that the event **Collide** occurs, then the adversary $\mathcal{A}$ can impersonate a valid user U with a legitimate conversation. The formula above follows by noticing that the hash function is second preimage resistant in BTCAS.

***Game*$_2$**: The game aborts if $\mathcal{A}$ does not succeed to make the event **Forge** happen.

Forge: Before makes the query of Corrupt(U), the adversary $\mathcal{A}$ makes an oracle query Send(π_U^i, m) in which the message m contains a valid, new message/signature pair for the public key of the user U. Therefore,

$|Adv_2(\mathcal{A}) - |Adv_1(\mathcal{A})| < negl(\lambda)$.

Proof. Supposing that the event **Forge** occurs, the algorithm $\mathcal{F}$ can be constructed to output a forgery against the digital signature with a non-negligible advantage. The formula above follows because the digital signature algorithm is existentially unforgeable under an adaptive chosen-message attack in our BTCAS.

***Game*$_3$**: The game aborts if the adversary $\mathcal{A}$ can break the security of blockchain that is to tamper with user data stored on the blockchain. The security of blockchain yields the following formula.

$|Adv_3(\mathcal{A}) - |Adv_2(\mathcal{A})| < negl(\lambda)$.

Concluding the proof of Lemma 1. Supposing that $Adv_3(\mathcal{A})$ above cannot be ignored, we can construct a blockchain adversary $\mathcal{D}$ which breaks the security of blockchain as follows: $\mathcal{D}$ executes all the games for $\mathcal{A}$. In response to Execute, $\mathcal{D}$ finds the public key with the same hash as the user's public key. In response to Send, $\mathcal{D}$ creates a legitimate signature. Then, $\mathcal{D}$ can tamper with user data stored on the blockchain as mentioned in ***Game*$_3$**. This is inconsistent with the security of the blockchain, thus the advantage of $\mathcal{A}$ is negligible.

Lemma 2. Suppose the blockchain network is secure. Then, the protocol Π_2 executed between server and responder is a TP-U-EA secure protocol.

Proof. The security of the blockchain yields the Lemma 2.

Theorem 1. Suppose the hash function is second preimage resistant, the digital signature algorithm is existentially unforgeable under an adaptive chosen-message attack and the blockchain network is secure. Then, the BTCAS is a secure unilateral authentication protocol.

Proof. Let the random number N be the identifier of the sessions, by **Lemma 1** and **Lemma 2, Theorem 1** holds.

There is a problem we must emphasize. By default, the data stored in the blockchain is public and visible to all members who belong to the channel, which is a risk of privacy disclosure. To protect the privacy of user data, we only save the hash value into the blockchain instead of the data itself. The identity data of the user stored in blockchain have the privacy property since the hash function integrated into BTCAS is preimage resistant.

6. Efficiency analysis

In this section, we first make a comparison between our BTCAS scheme and some other schemes proposed before in terms of Universal, TTP-Independence, and No Certificate Management. Then we discuss the implementation details and evaluate the performances of our scheme deployed on Hyperledger Fabric.

Table 3
Comparison of related protocols

Protocol	Type	Universal	TTP-Independence	No Certificate Management
Protocol in [4,11,12]	PKI	✗	√	✗
Protocol in [13-15]	IBE	✗	✗	√
Protocol in [5,18-20,23,24]	Blockchain	✗	√	✗
Protocol in [9]	Obfucation	√	✗	√
Our Protocol	Blockchain	√	√	√

6.1. Comparison with related schemes

In this subsection, we compare our BTCAS scheme with some typical PKI-Based works, IBE-based works, Blockchain-based works, and a universal work from many aspects, including Universal, which means this scheme allows different domains adopt various cryptographic settings, TTP-Independence, which tells us whether the scheme relies on TTP or not, No Certificate Management, which shows that whether the scheme involved in certificate management.

As shown in Table 3, only our BTCAS scheme supports all the features. The works based on traditional PKI have the burden of certificate management and IBE-based works necessitate the TTP (e.g., PKG) to generate private keys. And more, they require all participants to adopt a common cryptographic setting, which means that these protocols are "incomplete cross-domain ". Blockchain-based schemes further improve the efficiency of the authentication process and can get rid of the introduction of TTP, but these schemes are still not universal. Using the indistinguishability obfuscation as the main tool, Lan et al. [9] proposed a one-round cross-domain group key exchange protocol that can support cross-domain interaction from different cryptographic settings, however, a TTP needs to be introduced to choose a random key for the globally agreed domain parameter at the beginning of the protocol. In addition to blockchain, our BTCAS also takes advantage of the Smart Contract (chaincode in Hyperledger), and thus achieves the best performance.

6.2. Experiment performance

In terms of computation overhead, we implemented a prototype of our BTCAS scheme to evaluate its performance. The experiment is conducted using a server with Intel(R) Xeon(R) CPUE5-2680v4@2.40GHz processor and 250 GB RAM, running CentOS 7.6(64 bit). We set up the blockchain network based on Fabric version 1.4.1, consisting of one orderer node and two organizations which maintain a different number of peer nodes (1-5) respectively.

We measure our BTCAS scheme depending on a blockchain performance benchmark framework, the Caliper [39], by modifying the parameters contained in configuration files. In the process of measurement, we focused on the property of throughput and average transaction latency by increasing the function request send rate. As a reference, we divide the operations involved in our chaincode into two types: Invoke operation and Query operation. The first is to create a new key-value pair data and store it into the blockchain network, which will change the ledger stored in every peer node belonging to the same channel (*PublishDomain* and *PublishUser* in BTCAS). The Query operation is to query the data stored in blockchain before, which obviously will not change the ledger (*Verify* in BTCAS).

System performance of Invoke operation: Considering 2 to 10 peer nodes, the throughput and average transaction latency of Invoke operation are shown in Figure 4(a) and Figure 4(b). In terms of different functions, we can easily see that the performance of the two functions is almost the same, even in different network environments. Figure 4(a) indicates that with the increase of send rate, the throughput rate increases linearly at the beginning, and then, the throughput rate tends to be steady with a maximum value. As shown in Figure 4(b), the average transaction latency keeps stable at first and then rises rapidly with the increase of send rate. After the send rate reaches a certain value, a higher send rate causes steady throughput and higher latency may indicate the server has a bottleneck. In terms of different network environments, Figure 4(a) and Figure 4(b) show that with the increase of the number of peer nodes, the maximum throughput decreases and the average transaction latency increases. As mentioned before, Invoke operation will change the ledger stored in every peer node belonging to the same channel which means the more nodes in the network, the more copies need to be synchronized. From Table 4, we can see the storage cost of function *PublishDomain* and *PublishUser* for different types of nodes. However, we need to emphasize that the security of the blockchain is based on a great deal of computing power and a large amount of full backup of ledger data, so its a trade-off between security and performance.

Table 4
Storage cost of 1000 transactions

Node type	Function		
	PublishDomain	PublishUser	Verify
Peer	6.6MB	6.7MB	0MB
Orderer	4.7MB	4.7MB	0MB

System performance of Query operation: As shown in Figure 4(c) and Figure 4(d), the throughput increases linearly with the increase of send rate all the time, and the average transaction latency increases slightly. The system performance of Query operation is largely consistent among the different networks. As shown in Table 4, the Query operation only queries the data stored in the blockchain instead of changing it, which means this type of transaction does not have the process of ordering and transaction synchronization between nodes, thus responses quickly.

By testing in the different network environment and configuration, it is clear that with the increase of the number of peer nodes, the performance of Invoke operation degrades, and the Query operation has stable and good performance. Because the system performance metrics of our scheme are mainly dependent on the blockchain platform Hyperledger Fabric, the experimental results are consistent with the results of [40] and [41] which conducted effective performance evaluations of Hyperledger Fabric. These works can be referred to get a more detailed performance of our scheme.

Since Initialization (*PublishDomain*in BTCAS) and Intra-domain authentication (*PublishUser* in BTCAS) is a one-off process in our BTCAS, we can omit this and focus on the cost of the phase of Cross-domain authentication (*Verify* in BTCAS) which has excellent performance. And we stress again that all the computation involved in authentication phase is completed by blockchain instead of the authentication server, the authentication server has almost no computational burden so that can provide authentication services for a large number of users.

7. Conclusion

Benefits from the technology of HyperLedger [30] with rich support of cryptography algorithm implementation, we realize a blockchain-based thoroughly cross-domain authentication scheme BTCAS which can achieve the goal of "thoroughly cross-domain" without introducing TTP. Our BTCAS scheme provides "thoroughly cross-domain" authentication for participants from different domains (with different cryptographic settings), which is more flexible for practical scenarios. Besides, with the security of the adopted underlying blockchain network, signature algorithm, and hash function, our scheme is provably secure in the standard model. We also implement our protocol on the top of Hyperledger and the evaluation result shows the efficiency of our scheme. We are currently considering integrating more algorithms like SM [42,43] algorithms into HyperLedger to further expand the application scenarios of our cross-domain authentication scheme. And for efficiency, a more efficient blockchain platform also needs to be proposed.

Declaration of Competing Interest

The authors declare that they do not have any financial or non-financial conflict of interests.

CRediT authorship contribution statement

Hongxia Zhang: Conceptualization, Software, Writing - original draft. **Xingshu Chen:** Supervision. **Xiao Lan:** Investigation, Writing - original draft, Writing - review & editing. **Hongjian Jin:** Software, Formal analysis, Validation. **Qi Cao:** Software, Validation.

Acknowledgments

This study is funded by the National Natural Science Foundation of China (Grant No.U19A2081, No.61802270, No.61802271) and the Fundamental Research Funds for the Central Universities (No. SCU2018D018, No. SCU2018D022, No. 2019SCU12069).

References

[1] Jain AK, Hong L, Pankanti S, Bolle R. An identity-authentication system using fingerprints. Proceedings of the IEEE 1997;85(9):1365–88.
[2] Hua-Xi P. An identity-based authentication model for multi-domain. Chinese Journal of Computers 2006;8(29):8.
[3] Kim H, Shin KG, Dabbous W. Improving cross-domain authentication overwireless local area networks. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05). IEEE; 2005. p. 127–38.
[4] Hwang J-B, Do-Woo K, Lee Y-K, Han J-W. Two layered pki model for device authentication in multi-domain home networks. In: 2006 IEEE International Symposium on Consumer Electronics. IEEE; 2007. p. 1–6.
[5] Dong G, Chen Y, Fan J, Bai J, Zhang P, Li F. Anonymous cross-domain authentication scheme for medical pki system. In: Proceedings of the ACM Turing Celebration Conference-China. ACM; 2019. p. 68.
[6] Wu L, Wang J, Choo K-KR, Li Y, He D. An efficient provably-secure identity-based authentication scheme using bilinear pairings for ad hoc network. Journal of information security and applications 2017;37:112–21.
[7] Chakraborty S, Raghuraman S, Rangan CP. A pairing-free, one round identity based authenticated key exchange protocol secure against memory-scrapers. IACR Cryptology ePrint Archive 2016;2016:354.
[8] Shamir A. Identity-based cryptosystems and signature schemes. In: Workshop on the theory and application of cryptographic techniques. Springer; 1984. p. 47–53.
[9] Lan X, Xu J, Guo H, Zhang Z. One-round cross-domain group key exchange protocol in the standard model. In: International Conference on Information Security and Cryptology. Springer; 2016. p. 386–400.
[10] Liu C, Feng Y, Fan M, Wang G. Pki mesh trust model based on trusted computing. In: 2008 The 9th International Conference for Young Computer Scientists. IEEE; 2008. p. 1401–5.
[11] Millán GL, Pérez MG, Pérez GM, Skarmeta AFG. Pki-based trust management in inter-domain scenarios. Computers & Security 2010;29(2):278–90.
[12] Zhang W, Wang X, Khan MK. A virtual bridge certificate authority-based cross–domain authentication mechanism for distributed collaborative manufacturing systems. Security and Communication Networks 2015;8(6):937–51.
[13] McCullagh N, Barreto PS. A new two-party identity-based authenticated key agreement. In: Cryptographers Track at the RSA Conference. Springer; 2005. p. 262–74.
[14] Cao X, Kou W, Du X. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. Information Sciences 2010;180(15):2895–903.
[15] Yuan C, Zhang W, Wang X. Eimakp: Heterogeneous cross-domain authenticated key agreement protocols in the eim system. Arabian Journal for Science and Engineering 2017;42(8):3275–87.
[16] Pilkington M. Blockchain technology: principles and applications. Research handbook on digital transformations. Edward Elgar Publishing; 2016.
[17] Clack CD, Bakshi VA, Braine L. Smart contract templates: essential requirements and design options. arXiv preprint arXiv:161204496 2016.
[18] Zhou Z, Li L, Li Z. Efficient cross-domain authentication scheme based on blockchain technology. Journal of Computer Applications [J] 2018;38(2):316–20.
[19] Liu D, Li D, Liu X, Ma L, Yu H, Zhang H. Research on a cross-domain authentication scheme based on consortium blockchain in V2G networks of smart grid. In: 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2). IEEE; 2018. p. 1–5.
[20] Wang W, Hu N, Liu X. Blockcam: A blockchain-based cross-domain authentication model. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). IEEE; 2018. p. 896–901.
[21] Wu L, Du X, Wang W, Lin B. An out-of-band authentication scheme for internet of things using blockchain technology. In: 2018 International Conference on Computing, Networking and Communications (ICNC). IEEE; 2018. p. 769–73.
[22] Bendiab K, Kolokotronis N, Shiaeles S, Boucherkha S. Wip: A novel blockchain-based trust model for cloud identity management. In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE; 2018. p. 724–9.
[23] Yao Y, Chang X, Mišić J, Mišić VB, Li L. Bla: Blockchain-assisted lightweight anonymous authentication for distributed vehicular fog services. IEEE Internet of Things Journal 2019;6(2):3775–84.
[24] Li C, Wu Q, Li H, Liu J. Trustroam: A novel blockchain-based cross-domain authentication scheme for wi-fi access. In: International Conference on Wireless Algorithms, Systems, and Applications. Springer; 2019. p. 149–61.
[25] Nakamoto S, et al. Bitcoin: A peer-to-peer electronic cash system; 2008.
[26] Ethereum W. A secure decentralised generalised transaction ledger [j]. Ethereum project yellow paper 2014;151:1–32.
[27] Hopwood D, Bowe S, Hornby T, Wilcox N. Zcash protocol specification. Tech rep 2016–110 Zerocoin Electric Coin Company, Tech Rep 2016.
[28] Khan MA, Salah K. Iot security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems 2018;82:395–411.
[29] Valdeolmillos D, Mezquita Y, González-Briones A, Prieto J, Corchado JM. Blockchain technology: A review of the current challenges of cryptocurrency. In: International Congress on Blockchain and Applications. Springer; 2019. p. 153–60.
[30] Cachin C. Architecture of the hyperledger blockchain fabric. In: Workshop on distributed cryptocurrencies and consensus ledgers, 310; 2016. p. 4.
[31] Buterin V, et al. A next-generation smart contract and decentralized application platform. white paper 2014;3:37.
[32] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things. Ieee Access 2016;4:2292–303.
[33] Mezquita Y, Valdeolmillos D, González-Briones A, Prieto J, Corchado JM. Legal aspects and emerging risks in the use of smart contracts based on blockchain. In: International Conference on Knowledge Management in Organizations. Springer; 2019. p. 525–35.
[34] Nasir Q, Qasse IA, Abu Talib M, Nassif AB. Performance analysis of hyperledger fabric platforms. Security and Communication Networks 2018;2018.
[35] Kennedy W, Ketelsen B, Martin ES. Go in Action; 2015.
[36] Pointcheval D, Stern J. Security proofs for signature schemes. In: International Conference on the Theory and Applications of Cryptographic Techniques. Springer; 1996. p. 387–98.
[37] Katz J, Lindell Y. Introduction to modern cryptography. Chapman and Hall/CRC; 2014.
[38] Brzuska C, Jacobsen H. A modular security analysis of EAP and IEEE 802.11. In: IACR International Workshop on Public Key Cryptography. Springer; 2017. p. 335–65.
[39] "Calipar". https://hyperledger.github.io/caliper//; Accessed: 2020-4-1.
[40] Kuzlu M, Pipattanasomporn M, Gurses L, Rahman S. Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability. In: 2019 IEEE International Conference on Blockchain (Blockchain). IEEE; 2019. p. 536–40.
[41] Jiang L., Chang X., Liu Y., Mii J., Mii V.B.. Performance analysis of hyperledger fabric platform: A hierarchical model approach.
[42] "Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves". http://www.sca.gov.cn/sca/xwdt/2010-12/17/1002386/files/b791a9f908bb4803875ab6aeeb7b4e03.pdf; Accessed:2020-4-1.
[43] "SM3 Cryptographic Hash Algorithm". http://www.sca.gov.cn/sca/xwdt/2010-12/17/1002389/files/302a3ada057c4a73830536d03e683110.pdf; Accessed: 2020-4-4.