A Privacy-Preserving and Auditable Scheme for Interfacing Public Blockchain with Consortium Blockchain

Zejun Lu[1] and Jiageng Chen[1,2(✉)]

[1] Wollongong Joint Institute, Central China Normal University, Wuhan, China
jiageng.chen@ccnu.edu.cn
[2] School of Computer Science, Central China Normal University, Wuhan, China

Abstract. With the development of blockchain technology, consortium blockchain is being applied in various scenarios. However, data and related assets are restricted to the closed consortium blockchain environment, and the end-users who do not belong to the consortium are difficult to gain access without extra authentication. Thus, architectures concerning cross-chain interaction appear, while most solutions have only limited functionalities. Moreover, few solutions have considered privacy from multiple perspectives, including the privacy of end-users, consortium members, or the data itself. This paper proposes a privacy-preserving and auditable architecture scheme for interfacing consortium blockchain members with end-users of the public blockchain. Our scheme enables end-users to communicate with the inner consortium in a verifiable, privacy-preserving, and auditable manner. More specifically, we improve the existing cross-chain network architectures to further protect the consortium members' privacy. Also, the communication and the transactions of the cross-chain interaction are protected and auditable. Concrete protocols are proposed, and security models and corresponding analyses are investigated.

Keywords: Blockchain · Cross-chain · Consortium · Privacy · Auditability

1 Introduction

Distributed ledger with globally agreed and immutable transaction history is made available through the invention of blockchain technology [28]. Blockchain-based smart contracts further expanded the flexibility of the blockchain with Ethereum [34] being one of the well-known examples.

New architectures of blockchains were designed to adapt to different scenarios, while decentralized applications based on blockchains have become a growing family. To facilitate the development of decentralized applications, many high-level programming languages used to implement smart contracts were developed,

W. Meng and W. Li (Eds.): BlockTEA 2022, LNICST 498, pp. 89–105, 2023.
https://doi.org/10.1007/978-3-031-31420-9_6

such as Solidity [31]. As a result, it has become much easier to implement complex functions on the blockchain. With the prevalence of the blockchain in various fields like medical [4], supply chain [36], cloud [40], financial [14], and many others, the fact that data flowing in the blockchain is transparent gives rise to the possibility of leaking sensitive data, which makes the systems vulnerable to many attacks [12]. Researchers have proposed many schemes to defend against those attacks. However, it is not enough because blockchain application scenarios are becoming more complex, and architectures with only public or consortium blockchains cannot meet sophisticated requirements in different environments. Therefore, cross-chain architectures are developed.

Cross-Chain Scenarios. For example, Virtuozzo [27] has a cloud federation platform consisting of many cloud service providers (CSPs), and the platform aims to provide cloud services to customers. The customers post their requests in a public network, and the request data flows from the public user network to the consortium network, which consists of the CSPs. Upon incoming a request from outside, CSPs in the closed network decide how to respond collaboratively, and then the response flows from the closed network to the public user network. Another example is the supply chain, products are produced by a collaboration of companies along the manufacturing chain that form a consortium network, and customers form a public network. Customers post their requests to trace the manufacturing process of a product in the public network, and then the requests are transferred to the closed network. The counterpart companies respond to the request, and the consortium authenticates the response before it is transferred to the public network and read by the corresponding user. Both public and consortium networks can be implemented by blockchain architectures, with a cross-chain data transfer mechanism to interface two networks. In this way, cross-chain communication can obtain good properties such as traceability and immutability, and the whole system can be decentralized. A regular cross-chain workflow generalized from application scenarios like cloud federation, supply chain, etc., consists of transactions, each of which can be summarized as follows:

1. An end-user U initiates a transaction by posting a request in the public blockchain.
2. The request is transferred from the public to the consortium blockchain.
3. The request is handled, and the consortium authenticates the response.
4. The response is transferred from the consortium to the public blockchain.
5. U reads the response with its authentication from the public blockchain.

Except for the transactions, the consortium may release some official announcements, the process of which is similar to steps 3 to 5.

Existing solutions have developed protocols to interface public and consortium blockchains, but their schemes either barely considered privacy or lack generality.

1.1 Related Work

Privacy-Preserving Schemes. In the financial field, assorted cryptographical tools are introduced and integrated into blockchain architectures to fulfill its strong requirement for privacy. Several cryptocurrencies featuring strong privacy attributes are developed as discussed follows. Earlier versions of Monero called CryptoNote [33] achieve its privacy by using Schnorr-style multilayered linkable spontaneous anonymous group signatures to authenticate the inputs of transactions, using Pedersen commitments to conceal the amounts, and using Bulletproofs [9] to prove the range is legitimate. Later versions eliminated trusted third parties (Group Manager) at the expense of anonymity revocation by using ring signatures. Ideas from mainly [32] and [3,24,25] construct the Monero running today. At the same time, [23,35,37] pay their attention on the privacy in consortium blockchain architectures. The privilege of the consortium auditor is used to develop auditing possibility, a feature essential in the real world to combat illegal practices such as money laundering. [18,22,29,38] investigates the future of the privacy-preserving cryptocurrencies. Cryptographical tools like anonymous credentials, homomorphic encryption, zero-knowledge range proof, etc., are used to realize their visions of privacy. The idea of threshold signature [6] plays an important role in the investigation history of privacy-preserving cryptocurrencies to cover the identity of users, and it will be introduced in our work as well.

Although privacy has been comprehensively studied, few schemes are extended to cross-chain architectures for public and consortium blockchains. A new scheme is needed because public blockchain architectures are not suitable for many scenarios, while consortium architectures barely provide a verifiable service to end-users outside the consortium.

Cross-Chain Schemes. Most papers conduct their research in the context of single blockchain architecture. While [2,21] favor developing the shards of blockchains, they have developed how to deal with transactions between different shards of blockchain. Recently, [17] has made progress in reducing the number of cross-shard transactions. [1] shows their interest in crossing two consortium blockchains by designing a transfer protocol with verifiability. [39] is concerned about how to communicate across distributed ledgers, while [5,26] focuses on the transfer of tokens and assets between public blockchains. Cross-chain schemes like [10] tried to share data safely by adding a higher layer to the blockchain, while [16] has further developed the cross-chain architecture by providing an interface between public and consortium blockchains, through which the data can be transferred safely, users can interact with another blockchain in a soundness manner.

However, the protocols in those schemes either lack interoperability and mobility of data between the consortium and public blockchains or barely consider the privacy of consortium members and end-users.

1.2 Our Contribution: Cross-Chain with Privacy and Auditability

This article proposes a generic privacy-preserving and auditable cross-chain scheme with the following properties.

- **Generality**: Our scheme can be implemented by an almost arbitrary combination of consortium blockchain and public blockchain, as long as they have robust consensus algorithms like *Proof of Work (PoW)* and supports smart contracts.
- **Privacy-Preserving**: The request and response in the transaction are concealed from other end-users, and the identity of consortium members is concealed from their peers when running protocols in our scheme.
- **Auditability**: The consortium auditor can audit suspicious transactions by revealing the response process and authentication process in the consortium.
- **Cross-chain Safeness**: Consortium members agree on the same order and set of incoming requests, while these requests can be traced in the public blockchain. The responses posted in the public blockchain can also be traced in the consortium blockchain.

2 Preliminaries

2.1 Mathematical Backgrounds

Our scheme demands three groups G_U, G_1 and G_2 to implement the cryptographical tools. G_U is a prime-order group used for a public key encryption algorithm. (G_{T1}, G_{T2}) are a (τ, t, ε)-bilinear group pair as defined in Definition 2.1 and we summarised it below:

Computational co-Diffie-Hellman(co-CDH) on (G_1, G_2). Given $g_2, g_2^x \in G_2$ and $h \in G_1$ as input, compute $h^x \in G_1$.

Decision co-Diffie-Hellman(co-DDH) on (G_1, G_2). Given $g_2, g_2^x \in G_2$ and $h, h^y \in G_1$ as input, output **yes** if $x = y$ and **no**, otherwise. (g_2, g_2^x, h, h^a) is called a co-Diffie-Hellman tuple when the answer is **yes**.

(τ, t, ε)-co-GDH group pair. Two groups (G_1, G_2) are a (τ, t, ε)-co-GDH group pair if the following properties are satisfied.

- The group operation on both G_1 and G_2 and the isomorphism map ψ from G_2 to G_1 can be computed in time at most τ.
- The co-DDH problem on (G_1, G_2) can be solved in time at most τ.
- No algorithm (t, ε) breaks co-CDH on (G_1, G_2).

Given a group pairs (G_1, G_2) as defined above and another group G_T s.t. $|G_1| = |G_2| = |G_T|$. A bilinear map is a map $e : G_1 \times G_2 \to G_T$ with the following properties:

- Bilinear: for all $u \in G_1, v \in G_2$, and $a, b \in \mathbb{Z}$, $e(u^a, u^b) = e(u, v)^{ab}$.
- Non-degenerate: $e(g_2, g_2) \neq 1$.

$((\tau, t, \varepsilon))$-bilinear group pair. Two order-p groups (G_1, G_2) are a $((\tau, t, \varepsilon))$-bilinear group pair if satisfy the following properties:

- The group operation on both G_1 and G_2 and the isomorphism map ψ from G_2 to G_1 can be computed in time at most τ.
- A group G_T of order p and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ exist, and e is computable in time at most τ.
- No algorithm (t, ε) breaks co-CDH on (G_1, G_2).

[19] indicates that with e, the problem of co-DDH can be efficiently solved. This solution constructs the verification part of the BLS signature [7]. The threshold signature algorithm we used in our scheme is built by applying the method of [6] on the BLS signature.

2.2 Threshold Signature

Consider a scenario, a message M is regarded as authenticated by the group only when more than t members out of n members have signed for M. We use BLS threshold signature [7] to achieve this effect. To introduce the BLS threshold signature $(TKeyGen, TSign, Verify)$, the BLS signature $(KeyGen, Sign, Verify)$ should be introduced first.

Let G_1 and G_2 be bilinear group pair where $|G_1| = |G_2| = p$ and view the hash function $H : \{0,1\}^* \rightarrow G_1$ as a random oracle.

- $(SK, PK) \leftarrow KeyGen(\lambda)$: As a PPT algorithm. On input a security parameter, generates a pair of secret and public keys.
- $\sigma \leftarrow Sign(M, SK)$: On input a secret key SK and a message M, generate the signature of M.
- $valid/invalid \leftarrow Verify(PK, M, \sigma)$: On input a public key PK, a message M, and a signature σ, return valid if the signature corresponds with the message given the public key, otherwise invalid.

Next, the t-out-of-n BLS threshold signature is introduced. The parameter n means how many users are there in the group having the right to sign, while t indicates at least how many shares of signatures are required to collect to reconstruct.

- $\{(SK_i, PK_i)\}^n \leftarrow TKeyGen(SK, PK, t, n)$: On input a secret key SK, a public key PK, two constant t and q where $t \leq q$, generates n shares of the secret key SK_i and public key PK_i.
- $\sigma_i \leftarrow TSign(M, SK_i)$: On input a share of secret key SK_i and a message M, generates a share of signature of M.

For clarity, two more functions are added for the BLS threshold signature.

- $valid/invalid \leftarrow TV(PK_i, SK_i, \sigma_i)$: On input of a share of PK_i, a share of secret key SK_i, a share of signature σ_i, return valid if the signature corresponds with the message given the public key, otherwise invalid.

– $\sigma \leftarrow TReconstruct(\{\sigma_i\}^t)$: On input t shares of signatures $\{\sigma_i\}^t$, a signature σ corresponds to the public key PK can be reconstructed.

However, we manage to limit the privilege of the consortium auditor. As a result of that, the process of threshold key generation $TKeyGen()$ is replaced by the method of [15]. The same outcome is acquired by running a protocol without trusted third parties. The protocol does not rely on the difficulty of DDH and thus can be used in our scheme [7].

3 System Model

In this section, the architecture of our system is initially introduced by describing different roles and their respective behaviors, and then follows the various attacks that our system may suffer, the design challenges we encountered, and the corresponding countermeasures against these obstacles.

3.1 System Architecture

The main objective of our system is to provide a mechanism through which any closed consortium can interface with the open network of end-users by the cross-chain architecture. In our vision, all the entities in our system need not worry about whom they should communicate with, and all the interactions are between the blockchain and themselves, through which an immutable history and possibility of auditing are provided, while a proper running of the system functions is ensured by implementing smart contracts. In particular, everyone can ask for a service or pieces of information from a consortium without joining it by posting their request in the public blockchain. Depending on the content of the request, the consortium members collaboratively generate and authenticate a response by posting their message in the consortium blockchain, respectively. According to [39], cross-chain communication requires trusted third party. Similar to the idea *Consensus on Consensus* of [16], it is reasonable that some consortium members are elected as trusted agents to bypass the requirement of trusted third parties. At last, to realize the auditability and preserve privacy, an auditor is needed in the consortium. However, the privileges of the auditor are limited. Malicious behaviors, such as forging a transaction that can be verified in the consortium blockchain, are hard to conduct for the auditor.

Accordingly, there are four types of entities in our scheme:

1. **End-users:** End-users are transaction initiators, requests from whom are mined by nodes running the consensus algorithm of the public blockchain. A transaction is initiated when a request is posted in the public blockchain. A transaction is completed when the corresponding response in the public blockchain is read and verified by the initiator.
2. **Consortium Members:** Consortium members are consortium blockchain nodes with their share of threshold signature secret key, running the consensus algorithm of the consortium blockchain. They generate responses or vote for the requests and respond by signing with their secret keys.

3. **Trusted Agents:** Trusted agents are public blockchain nodes as well as consortium blockchain nodes, running the consensus algorithms of both blockchains. They are responsible for transferring information between two blockchains. A trusted agent can be a consortium member at the same time.
4. **Auditor:** The auditor is a consortium blockchain node without a share of threshold signature secret key, running the consensus algorithm of the consortium blockchain. It is responsible for reconstructing the threshold signature and posting the reconstructed signature in the consortium blockchain. The reconstruction process can be revealed by it for auditing purposes when necessary.

3.2 Threats

Due to the data structure of the blockchain itself, systems based on blockchain may be vulnerable to several types of attacks. For a cross-chain architecture, an interface that safely transfers data between different blockchains should be developed. Meanwhile, the privacy of different entities from various perspectives should also be considered. The situations are discussed as follows, and our definitions of security and privacy are shown in Sect. 4.3.

Byzantine Behavior. It is assumed that at most t members out of all the n consortium members may exhibit some malicious actions [11]. Usually, the parameter t/n is less than 1/3. These t members may collaboratively manage to control the decision of the consortium. Besides, the parameters t and n are equal to the parameters of the same name in the threshold signature scheme in our scheme.

Sybil Attacks. *Sybil attacks* [13] denote attacks from those adversaries who can manipulate, imitate, or generate many identities to attack the system. In our system, before any request is processed by the consortium, it must have been mined by miners running the consensus algorithm of the public blockchain. For the fact that public blockchain can be joined without permission, consensus algorithms designed for public blockchains such as *PoW* and *Proof of Stake (PoS)* [20] have the feature of resisting *sybil attacks* natively. Therefore, whether our scheme can resist *sybil attacks* or not depends on the consensus algorithm of the chosen public blockchain.

Traceability. Traceability is one of the most important reasons we use blockchains. However, this characteristic should be further developed in cross-chain architectures. Our scheme focuses on the following requirements:

1. Consortium and public blockchain have their immutable transaction history separately, as they originally were.
2. Once a message is transferred from the public blockchain to the consortium blockchain; consortium members can find the corresponding history in the public blockchain.

3. Once a message is transferred from the consortium blockchain to the public blockchain; users can find the corresponding history in the consortium blockchain with the help of the auditor.

Interface Safety. Given the reality that a network in the real world cannot be completely synchronous, an upper bound is usually defined. A message with a transmission time more than that upper bound is considered missing and abandoned. However, the delay of unabandoned messages exists objectively and inevitably, which leads to the consequence that different users receive different sets of messages in different orders. Therefore, a safe cross-chain architecture requires consortium members to agree on which requests are transferred to the consortium and their specific order, without which confusion may be caused, and wrong responses may be returned to the customers. In particular, (1) Some consortium members may lose some mined blocks from the public blockchain. (2) Malicious members may reply to requests which have not been mined. (3) Conflicting requests from temporary forks [30] of the public blockchain. In our system, the elected consortium members compose the cross-chain interface, and the *First Signature*, which will be introduced in Sect. 4.1, ensures the consortium members have reached a consensus on the incoming requests.

We use the definition **Consortium Interface Safety** from [16] to define our safety as shown in Sect. 4.3.

Privacy. The privacy of the following entities is considered.

1. **End-users:** The request information may include sensitive information like invoices, credentials, etc. Given that the data on the public blockchain is available to everyone, information may be collected and used to identify the users. Moreover, in some scenarios, the request information contains no authentication information(like cloud federation providing cloud service to everyone), and the end customer is unwilling to let consortium members know which requests they have sent before. These problems can be addressed by public key encryption.
2. **Consortium Members:** Under the assumption that the manager is not corrupted, the identities of the consortium members should be concealed during the process of agreeing on an exact order of requests. This is achieved by threshold signature and public key encryption.

In our system, adversaries can read the information in the public and consortium blockchain, to corrupt consortium members or end-users. He may try to reveal a vote, forge a request or response with proper authentication, and reveal the advocators in a vote. Detailed security attributes we achieved are discussed in Sect. 4.3. Besides, our scheme does not consider network-level privacy issues, such as analyzing the data packets, network flows, or IP addresses.

4 Our Proposed Scheme

In this section, the workflow of a transaction is described first, which is the most used protocol in our scheme. Next, a comprehensive description of the whole scheme is given, and a security discussion follows.

4.1 Overview

A general transaction workflow includes the following three phases:

Initiate Transaction. This part describes the transaction initiation in the public blockchain.

1. An end-user seeks a service or some information. She sends her request with a user public key to the public blockchain miners.
2. Miners of the public blockchain collect the requests and public keys and pack them into a block, and then the block is committed in the public blockchain.
   - During this process, the *Blocknumber* of the block in the public blockchain and the *offset* of requests in this block are determined. *Blocknumber* and *offset* compose the unique order information of each request and determine the order in which requests are scheduled in the consortium.
3. Trusted agents (as a public blockchain node) are triggered upon a new block being committed. They verify it by methods like *Simplified Payment Verification (SPV)* and wrap the requests, user public keys, and their order information as consortium requests.

First Signature. This part describes the generation of the *First Signature*, which is not only for deciding the order of the consortium requests but also can vote on whether the consortium will schedule the request or not when necessary. After the *First Signature* is generated, requests can be scheduled and responded to.

1. Trusted agents (as a consortium blockchain node) post the consortium requests in the consortium blockchain by running the consensus algorithm.
2. The consortium members continuously read information from the consortium blockchain and sign the consortium requests. The consortium request, signature, and threshold public key are wrapped and encrypted by the auditor public key and posted in the consortium blockchain.
   - Here, the signature is a share of a threshold signature, which cannot prove that the consortium has admitted the consortium request. Only requests with a reconstructed signature, which can be verified by the consortium's unique threshold public key, are regarded as having achieved a consensus of the consortium and scheduled.

3. The auditor maintains a hash map to count the threshold signatures for each consortium request. Once the number of signatures for a request meets the threshold, the auditor reconstructs the signature and posts the reconstructed signature with the consortium request in the consortium blockchain. If the manager is given the right to assign tasks, she may designate a consortium member to respond.

It depends on the specific environment for whose response will be transferred to the public blockchain. Members can vote on the response content if the response is not encrypted. If the response is encrypted by the user's public key, members can vote on the identity of the responder. Maybe he is a certified expert in the field related to this request, the counterpart manufacturer, or the only one who knows the answer to the request, etc. Essentially, only the first response that completes the *Second signature* will be mined and committed in the consortium blockchain.

Second Signature. Up to here, there may be several [Consortium Request, First Signature] tuples in the consortium blockchain. To convince the end-users that the consortium has acknowledged the response, a signature representing the consortium must be attached to the response. This signature can be easily generated by off-chain multi-signature [6]. However, this approach is detrimental to the privacy and traceability of the system. We propose the *Second Signature* here, the process of which is very similar to steps 2,3 in *First Signature*.

1. Consortium members respond to the requests by posting their responses in the consortium blockchain, and the responses can be encrypted by the user public key when necessary.
2. Consortium members sign the response with their share of the threshold secret keys and encrypt their signature with the response by auditor public key, then post it in the consortium blockchain.
3. The auditor reconstructs the signature and posts the response with the signature in the consortium blockchain.
4. Trusted agents transfer the response with the corresponding signature to the public blockchain.

The proposed scheme is demonstrated in Fig. 1 and only the main parameters are shown. Notice that the *Second Signature* is omitted because its process is similar to the *First Signature*. A detailed description of the process can be found in Sect. 4.2.

Auditability. In the process described above, the auditor can reveal which consortium members have voted for which requests or responses. With the cooperation of the end-users, the auditor can reveal the encrypted response. In addition, the auditor can further corroborate his words by decrypting the corresponding record in the consortium blockchain.

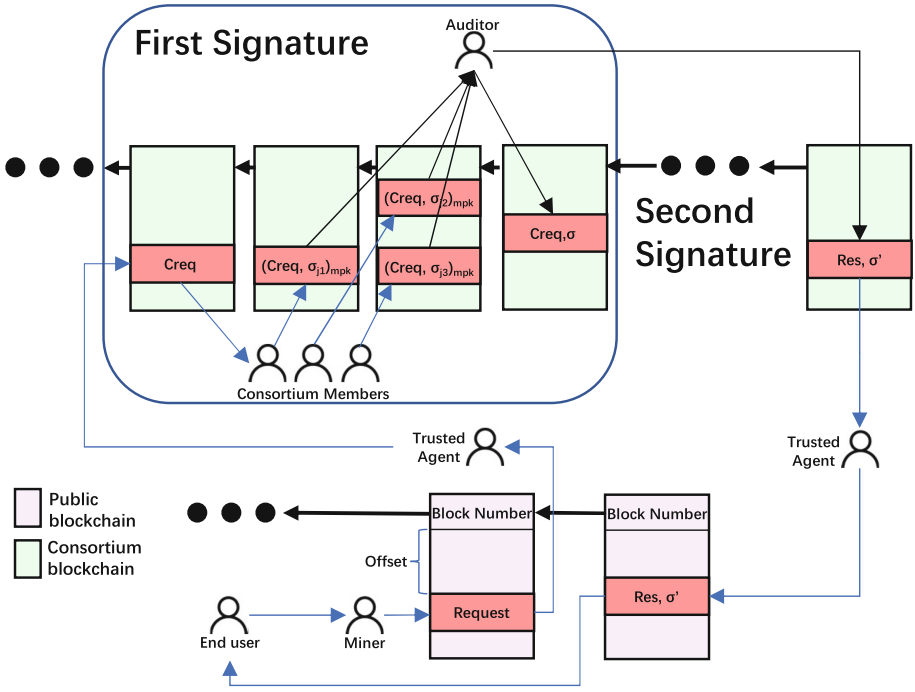

Fig. 1. The proposed scheme

4.2 Detailed Description

This section shows our detailed scheme, followed by the security and privacy theorem, with their proof sketch discussed.

Setup:

- Let G_1 and G_2 be a bilinear group pair conforms to Definition 2.1 with generators $g_1 \in G_1$ and $g_2 \in G_2$. $e : G_1 \times G_2 \rightarrow G_T$ be a bilinear map. The method to generate this type of group pair with proper parameters is introduced in [7,19].
- The auditor generates her keys for public key encryption (mpk, msk) and creates a hashmap $SignMap$ with $(Key, Values) = (Request, Signatures)$.
- End-users generate their keys for public key encryption (upk_i, usk_i).
- Consortium members run the none-trusted-party version $TKeyGen()$ with the other, achieve their share of secret key tsk_j and public key $tpk_j \leftarrow g_2^{tsk_j}$, publish their union public key tpk.
- Hash function $H : \{0,1\}^* \rightarrow G_1$ as a random oracle.

Functions:

1. $(v, x) \leftarrow KeyGen() : x \xleftarrow{R} \mathbb{Z}_p, v \leftarrow g_2^x$.
2. $\sigma \leftarrow Sign(M, x)$: $h \leftarrow H(M)$ and $\sigma \leftarrow h^x$.
3. $valid/invalid \leftarrow Verify(v, M, \sigma)$: $h \leftarrow H(M)$ and verify (g_2, v, h, σ) is a valid co-Diffie-Hellman tuple i.e. $e(h, v) \stackrel{?}{=} e(\sigma, g_2)$. The signature is valid if and only if the equation holds.
4. $\{(v_i, x_i)\}^n \leftarrow TKeyGen(x, v, t, n)$: A central authority picks a random polynomial $\omega \in \mathbb{Z}_p[X]$ of degree at most t-1 s.t. $\omega(0) = x$. For each user i, the authority gives her $x_i = \omega(i)$, its share of the secret key. The authority publishes v and n values $u_i = g_2^{x_i}$
   - Method of [15] is used to eliminate central authority and achieves the same outcome.
5. $\sigma_i \leftarrow TSign(M, x_i)$: $\sigma_i \leftarrow H(M)^{x_i}$.
6. $valid/invalid \leftarrow TVerify(v_i, M, \sigma_i)$: $h \leftarrow H(M)$, $e(h, v_i) \stackrel{?}{=} e(\sigma_i, g_2)$
7. $\sigma \leftarrow TReconstruct(\{\sigma_i, u_i\}^t)$: Each pair of (σ_i, u_i) is verified by $TVerify()$, then $\sigma \leftarrow \prod_{i=1}^{t} \sigma_i^{\lambda_i}$, $\quad where \quad \lambda_i = \frac{\prod_{j=1, j\neq i}^{t} -j}{\prod_{j=1, j\neq i}^{t} i-j} \pmod p$.
8. $toP(M)$: Message M is posted in the public blockchain.
9. $toC(M)$: Message M is posted in the consortium blockchain.

   And a public key encryption scheme:
10. $M_{pk} \leftarrow Enc(pk, M)$ and $M \leftarrow Dec(sk, M_{pk})$
11. $SPV(B)$: Simplified Payment Verification to B.

Publish Consortium Announcements. Sometimes the consortium makes some announcement, and the announcement represents the group's will. Thus it should attach a signature verifiable by tpk and has a record in the consortium blockchain.

1. A member j draft an announcement ANC_j.
   - $\sigma_j \leftarrow Tsign(ANC_j, tsk_j)$
   - $[ANC_j, \sigma_j]_{mpk} \leftarrow Enc(mpk, [ANC_j, \sigma_j])$
   - $toC([ANC_j, \sigma_j]_{mpk})$
   - $toC(ANC_j)$
2. The other members such as j_2 read and advocate ANC_j.
   - $\sigma_{j2} \leftarrow Tsign(ANC_j, tsk_{j2})$
   - $[ANC_j, \sigma_{j2}]_{mpk} \leftarrow Enc(mpk, [ANC_j, \sigma_{j2}])$
   - $toC([ANC_j, \sigma_{j2}]_{mpk})$
3. As soon as the auditor has collected $[ANC_j, \sigma_{j2}]_{mpk}$.
   (Status: The auditor lacks 1 signature to generate the *First Signature* of ANC_j and has not collected $[ANC_j, \sigma_{j2}]_{mpk}$.)
   - $[ANC_j, \sigma_{j2}] \leftarrow Dec(msk, [ANC_j, \sigma_{j2}]_{mpk})$
   - $valid \leftarrow Tverify(tpk_{j2}, ANC, \sigma_{j2})$
   - $\sigma \leftarrow TReconstruct(\sigma_{1\ldots t}, tpk_{1\ldots t})$
   - $toC(ANC_j, \sigma)$
4. Trusted agents read (ANC_j, σ).
   - $toP((ANC_j, \sigma)$

Transactions. Three phases of a transaction are described separately as follows:
Initiate Transaction

1. An end-user i seeks services or information.
   – $toP([Request_i, upk_i])$
2. Trusted agents read the request.
   – Define $ID_i = [Blocknumber_i, Offset_i]$, where $Offset_i$ uniquely identify $Request_i$ in the block.
   – $SPV([Request_i, upk_i, ID_i])$
   – Define $Creq_i = [Request_i, upk_i, ID_i]$

First Signature are denoted as σs.

1. Trusted agents.
   – $toC(Creq_i)$
2. Consortium members j_2 reads $Creq_i$ and advocates it.
   – $\sigma_{j2} \leftarrow Tsign(Creq_i, tsk_{j2})$
   – $[Creq_i, \sigma_{j2}, tpk_{j2}]_{mpk} \leftarrow Enc(mpk, [Creq_i, \sigma_{j2}, tpk_{j2}])$
   – $toC([Creq_i, \sigma_{j2}, tpk_{j2}]_{mpk})$
3. The auditor, reads $[Creq_i, \sigma_{j2}, tpk_{j2}]_{mpk}$.
   (Status: The auditor lacks 1 signature to generate the *First Signature* of $Creq_i$ and has not collected $[Creq_i, \sigma_{j2}, tpk_{j2}]_{mpk}$.)
   – $[Creq_i, \sigma_{j2}, tpk_{j2}] \leftarrow Dec(msk, [Creq_i, \sigma_{j2}, tpk_{j2}]_{mpk})$
   – $valid \leftarrow TVerify(tpk_{j2}, Creq_i, \sigma_{j2})$
   – $\sigma \leftarrow TReconstruct(\sigma_{1\ldots t}, tpk_{1\ldots t})$
   – $ToC(Creq_i, \sigma)$

Second Signature are denoted as σ's.

1. Consortium member j_3 responds to $Creq_i$.
   – $Verify(tpk, Creq_i, \sigma)$
   – Generate the response Res_{j3}.
   – If required, $Res_{j3} \leftarrow Enc(upk_i, Res_{j3})$.
   – $toC(Res_{j3})$
   – $\sigma'_{j3} \leftarrow Tsign(Res_{j3}, tsk_{j3})$
   – $[Res_{j3}, \sigma'_{j3}, tpk_{j3}]_{mpk} \leftarrow Enc(mpk, [Res_{j3}, \sigma'_{j3}, tpk_{j3}])$
   – $toC([Res_{j3}, \sigma'_{j3}, tpk_{j3}]_{mpk})$
2. Consortium member j_4 reads Res_{j3} and advocates it.
   – $\sigma'_{j4} \leftarrow Tsign(Res_{j3}, tsk_{j4})$
   – $[Res_{j3}, \sigma'_{j4}, tpk_{j4}]_{mpk} \leftarrow Enc(mpk, [Res_{j3}, \sigma'_{j4}, tpk_{j4}])$
   – $toC([Res_{j3}, \sigma'_{j4}, tpk_{j4}]_{mpk})$
3. The auditor, reads $[Res_{j3}, \sigma'_{j4}, tpk_{j4}]_{mpk}$.
   (Status: The auditor lacks 1 signature to generate the *Second Signature* of Res_{j3} and has not collected $[Res_{j3}, \sigma'_{j4}, tpk_{j4}]_{mpk}$.)
   – $[Res_{j3}, \sigma'_{j4}, tpk_{j4}] \leftarrow [Res_{j3}, \sigma'_{j4}, tpk_{j4}]_{mpk})$
   – $valid \leftarrow TVerify(tpk_{j4}, Res_{j3}, \sigma'_{j4})$
   – $\sigma \leftarrow TReconstruct(\sigma_{1\ldots t}, tpk_{1\ldots t})$
   – $ToC(Res_{j3}, \sigma')$
4. Trusted agents reads Res_{j3}, σ'.
   – $ToP(Res_{j3}, \sigma')$

Audit. The manager may reveal a row in her $SignMap$ and decrypt the corresponding history in the consortium blockchain when a request is considered suspicious. The response record in the consortium and public blockchain can be revealed if the user's secret key is provided.

4.3 Security Discussion

We discuss the security definitions as follows, and formal definitions and proofs will be shown in the full version of this paper.

Definition 1 (q-Consortium Soundness). *A cross-chain interaction scheme has q-Consortium Soundness if for all PPT adversary A, who is allowed to corrupt less than* $(q \cdot |consortium|)$ *consortium members, corrupt the manager, cannot produce a valid and verifiable response to a request without honest consortium member signing it.*

Theorem 1. *Our scheme has q-Consortium Soundness if the underlying t-out-of-n threshold signature algorithm is unforgeable, where* $q = t/n$.

Generally, a secure t-out-of-n threshold signature algorithm ensures that any PPT adversary, who is allowed to corrupt up to t users, cannot produce a valid signature(not one's share of signature). An adversary who can attack our scheme will generate a valid response, which contains a signature. This signature can be used to attack the unforgeability of the underlying threshold signature algorithm.

Definition 2 (Consortium Interface Safety). *The interface should ensure that all the correct consortium members agree on the same set of incoming consumer requests in the same order.*

Theorem 2. *Our scheme has Consortium Interface Safety if the underlying threshold signature is secure and the consensus algorithm of the chosen public blockchain provides a determined sequence of requests.*

Intuitively, the process of generating the *First Signature* is simultaneously deciding the order of requests.

Definition 3 (q-Consortium Member Privacy). *A cross-chain interaction scheme has q-Consortium Member Privacy if for all PPT adversary A, who is allowed to corrupt less than* $(q \cdot |consortium|)$ *consortium members, cannot give a correct list of consortium members advocating for a given request.*

Theorem 3. *Our scheme has q-Consortium Member Privacy if the underlying public key encryption algorithm is CPA secure and the underlying secret sharing scheme of the threshold signature scheme is secure.*

The definition of a secure *secret sharing* scheme and the proof of *Shamir's secret sharing scheme is secure* can be found in [8]. An adversary who can attack our scheme will generate a list of consortium members and a request, which can be used to attack the security of Shamir's secret sharing scheme.

5 Conclusion

By redesigning the protocols with various cryptographic tools, we enhanced the privacy of the consortium members by concealing their identities from the voting process. Also, we provided the auditability to the cross-chain architecture by revealing suspicious transactions. Our future work includes formal security proof as well as the evaluation of the scheme in a simulated scenario.

Acknowledgement. This work has been partly supported by the Fundamentral Research Funds for the Central Universities (No. 30106220482).

References

1. Abebe, E., et al.: Enabling enterprise blockchain interoperability with trusted data transfer (industry track). In: Proceedings of the 20th International Middleware Conference Industrial Track, pp. 29–35 (2019)
2. Al-Bassam, M., Sonnino, A., Bano, S., Hrycyszyn, D., Danezis, G.: Chainspace: a sharded smart contracts platform. arXiv preprint arXiv:1708.03778 (2017)
3. Au, M.H., Chow, S.S.M., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 101–115. Springer, Heidelberg (2006). https://doi.org/10.1007/11774716_9
4. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: MedRec: using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD), pp. 25–30. IEEE (2016)
5. Bentov, I., Ji, Y., Zhang, F., Breidenbach, L., Daian, P., Juels, A.: Tesseract: real-time cryptocurrency exchange using trusted hardware. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1521–1538 (2019)
6. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_3
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004)
8. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)
9. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 315–334. IEEE (2018)
10. Cash, M., Bassiouni, M.: Two-tier permission-ed and permission-less blockchain for secure data sharing. In: 2018 IEEE International Conference on Smart Cloud (SmartCloud), pp. 138–144. IEEE (2018)
11. Castro, M., Liskov, B., et al.: Practical Byzantine fault tolerance. In: OsDI, vol. 99, pp. 173–186 (1999)
12. Chainalysis: The 2020 state of crypto crime. https://go.chainalysis.com/rs/503-FAP-074/images/2020-Crypto-Crime-Report.pdf. Accessed 24 Sept 2022
13. Douceur, J.R.: The Sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_24

14. Fuchsbauer, G., Orrù, M., Seurin, Y.: Aggregate cash systems: a cryptographic investigation of mimblewimble. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 657–689. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_22
15. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_21
16. Ghosh, B.C., Bhartia, T., Addya, S.K., Chakraborty, S.: Leveraging public-private blockchain interoperability for closed consortium interfacing. In: IEEE Conference on Computer Communications, IEEE INFOCOM 2021, pp. 1–10. IEEE (2021)
17. Huang, H., et al.: Brokerchain: a cross-shard blockchain protocol for account/balance-based state sharding. In: IEEE INFOCOM (2022)
18. Jivanyan, A.: Lelantus: towards confidentiality and anonymity of blockchain transactions from standard assumptions. IACR Cryptology ePrint Archive **2019**, 373 (2019)
19. Joux, A., Nguyen, K.: Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. J. Cryptol. **16**(4), 239–247 (2003)
20. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
21. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: OmniLedger: a secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598. IEEE (2018)
22. Lai, R.W., Ronge, V., Ruffing, T., Schröder, D., Thyagarajan, S.A.K., Wang, J.: Omniring: scaling private payments without trusted setup. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–48 (2019)
23. Li, W., Sforzin, A., Fedorov, S., Karame, G.O.: Towards scalable and private industrial blockchains. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 9–14 (2017)
24. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_28
25. Liu, J.K., Wong, D.S.: Linkable ring signatures: security models and new schemes. In: Gervasi, O., et al. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005). https://doi.org/10.1007/11424826_65
26. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. Cryptology ePrint Archive (2018)
27. Morgan, J.: Virtuozzo. https://www.virtuozzo.com/. Accessed 24 Sept 2022
28. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Business Review, p. 21260 (2008)
29. Noether, S., Goodell, B.: Triptych: logarithmic-sized linkable ring signatures with applications. In: Garcia-Alfaro, J., Navarro-Arribas, G., Herrera-Joancomarti, J. (eds.) DPM/CBT -2020. LNCS, vol. 12484, pp. 337–354. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66172-4_22

30. Shahsavari, Y., Zhang, K., Talhi, C.: A theoretical model for fork analysis in the bitcoin network. In: 2019 IEEE International Conference on Blockchain (Blockchain), pp. 237–244. IEEE (2019)
31. Soliditylang.org: Solidity. https://soliditylang.org/. Accessed 24 Sept 2022
32. Sun, S.-F., Au, M.H., Liu, J.K., Yuen, T.H.: RingCT 2.0: a compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10493, pp. 456–474. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66399-9_25
33. Van Saberhagen, N.: Cryptonote v 2.0 (2013)
34. Wood, G., et al.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**(2014), 1–32 (2014)
35. Wüst, K., Kostiainen, K., Čapkun, V., Čapkun, S.: PRCash: fast, private and regulated transactions for digital currencies. In: Goldberg, I., Moore, T. (eds.) FC 2019. LNCS, vol. 11598, pp. 158–178. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32101-7_11
36. Xu, X., Rahman, F., Shakya, B., Vassilev, A., Forte, D., Tehranipoor, M.: Electronics supply chain integrity enabled by blockchain. ACM Trans. Des. Autom. Electron. Syst. (TODAES) **24**(3), 1–25 (2019)
37. Yuen, T.H.: PAChain: private, authenticated & auditable consortium blockchain and its implementation. Futur. Gener. Comput. Syst. **112**, 913–929 (2020)
38. Yuen, T.H., et al.: RingCT 3.0 for blockchain confidential transaction: shorter size and stronger security. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 464–483. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_25
39. Zamyatin, A., et al.: SoK: communication across distributed ledgers. In: Borisov, N., Diaz, C. (eds.) FC 2021. LNCS, vol. 12675, pp. 3–36. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-662-64331-0_1
40. Zhou, H., Ouyang, X., Ren, Z., Su, J., de Laat, C., Zhao, Z.: A blockchain based witness model for trustworthy cloud service level agreement enforcement. In: IEEE Conference on Computer Communications, IEEE INFOCOM 2019, pp. 1567–1575. IEEE (2019)