PAChain: Private, Authenticated and Auditable Consortium Blockchain

Tsz Hon Yuen(✉)

The University of Hong Kong, Pok Fu Lam, Hong Kong
thyuen@cs.hku.hk

Abstract. Blockchain provides a distributed ledger recording a globally agreed, immutable transaction history, which may not be suitable for Fintech applications that process sensitive information. This paper aims to solve three important problems for practical blockchain applications: privacy, authentication and auditability.

Private transaction means that the transaction can be validated without revealing the transaction details, such as the identity of the transacting parties and the transaction amount. Auditable transaction means that the complete transaction details can be revealed by auditors, regulators or law enforcement agencies. Authenticated transaction means that only authorized parties can be involved in the transaction. In this paper, we present a private, authenticated and auditable consortium blockchain, using a number of cryptographic building blocks.

Keywords: Blockchain · Privacy

1 Introduction

Blockchain is a fast-growing field of technology since it is recognized as the core component of Bitcoin [10]. Blockchain can serve as a distributed ledger in a peer-to-peer network to provide publicly verifiable data. Blockchain can be mainly classified into three categories: public, private and consortium blockchain. The *public blockchain* is an open, permissionless system such that every one is allowed to join as a user or miner freely. Bitcoin and most cryptocurrencies falls into this category. Public blockchain is suitable for decentralized network without trusted authority. However, they usually have limited throughput. The *private blockchain* is a closed, permissioned system such that a user must be validated in order to use the system. It is a traditional centralized system with auditability attached. The *consortium blockchain* is a partly private blockchain. The submission of transactions can be performed by many (authorized) users, but the verification of transactions is only permitted by a few predetermined parties. Consortium blockchain provides a higher efficiency (more than 10 K transactions per second (tps)) than the public blockchain, without consolidating power with a single party as the private blockchain. Consortium blockchain is suitable for organizational collaboration.

Consortium blockchain received a great interest from the industry due to the similarity with the existing business model, especially in the finance industry. Hyperledger, an umbrella project of open source consortium blockchain, has 130 members including members from the IT industry (e.g., IBM, Intel) and the financial industry (e.g., JP Morgan, American Express).

Privacy in Blockchain. Privacy is important for commercial system, especially in the financial sector where money is transferred from one party to another. No one would like to have his bank account transaction history posted on a public blockchain. We define three key privacy properties that we want to achieve in this paper:

1. Sender Privacy: the sender's identity is not known by any third party and two valid transactions of the same sender should not be linked.
2. Recipient Privacy: the recipient's identity is not known by any third party and two valid transactions of the same recipient should not be linked.
3. Transaction Privacy: the content of the transaction is not known by any third party. General transaction privacy for smart contract is difficult to achieve without using general zero-knowledge proof of circuit or fully homomorphic encryption, which are both not quite practical. In this paper, we only consider the privacy for *transaction amount*.

The above conditions should hold for any third party (including the parties running the consensus algorithm). Cryptocurrencies such as Monero and Zcash offer privacy in the public blockchain. There are also a number of academic and industrial solutions for privacy-preserving consortium blockchain. Details will be discussed in Sect. 3.1.

Our Contributions: Privacy, Authenticated and Auditable in Consortium Blockchain. Auditability is essential for financial blockchain applications and unconditional anonymity may not be desirable. Financial institution has to undertake both internal and external audit for checking if there is any money laundering or terrorist-related activities, under regulations from the government. Auditing is usually performed by sample checking of all transaction records. In case of court order, the institute has to provide the complete information of a particular transaction to the court. For all of the above situations, the privacy of certain transaction has to be revoked if necessary. For simplicity, we denote the party to legally revoke privacy as the *auditor*.

Authentication is important for consortium blockchain in two aspects. First, the consortium companies need to ensure that the user is authenticated to use the system (e.g., he has paid/subscribed for the blockchain service). The consortium companies do not earn from the "mining" process and no new coin is generated from consortium blockchain. Second, authentication is useful for tracing real user identity during the auditing process. If users can transact in the consortium blockchain without registration, then the auditor can only discover the self-generated public key after opening the transaction. The real world identity can only be recovered if the user was registered before and is authenticated during the transaction.

Table 1. Comparison with existing privacy-preserving blockchain schemes

		Sender privacy	Recipient privacy	Transaction privacy	Auditability	Authentication	Efficiency
Public blockchain	Monero/RingCT-based solutions	◕	●	●			◐
	Zcash/zk-SNARK-based solutions	●	●	●			◔
	DAP [8]	●	●	●	●		◔
Consortium/ Private blockchain	R3 Corda, [9], Hyperledger Fabric	◔	◔	◔		●	●
	Fabric experiment	◐	◔	◔	◔	◐	◕
	PRCash [15]	◐	◐	●	◐		◕
	This paper	◕	●	●	●	●	◕

In this paper, we show how to construct a private, authenticated and auditable consortium blockchain: PAChain. We give the sender privacy, recipient privacy and transaction privacy by three separate modules. Auditability is provided for all three modules. Authentication is analyzed for the sender privacy and recipient privacy modules. It allows us to analysis the security of each module clearly. It gives the flexibility for system architects to choose the properties according to the business requirements. Therefore, our construction is suitable to be deployed in real world business use cases. In our construction, we use a number of cryptographic techniques (e.g., anonymous credential, zero-knowledge range proof, additive homomorphic encryption) and modify them for higher efficiency in consortium blockchain. Table 1 gives the comparison of our paper and related works described in Sect. 3.1.

2 PAChain Overview

In this paper, we show the high level overview of how to achieve privacy and auditability in consortium blockchain.

2.1 System Model

In blockchain system, clients can submit transactions (Tx) to nodes running the consensus algorithms. The nodes validate the Tx and add it to a block. In this paper, we extend the system model of Hyperledger Fabric 1.0[1], which is designed for consortium blockchain. Transactions have to be *endorsed* by endorsers and only endorsed Txs can be committed.

[1] Hyperledger Fabric Architecture Explained. http://hyperledger-fabric.readthedocs.io/en/latest/arch-deep-dive.html.

There are five types of nodes in PAChain:

1. Client: A client invokes a Tx to the endorsers, and also broadcasts the transaction-proposals to the orderer.
2. Peer: There are two types of peers. *Endorser* checks the validity of the Tx submitted by the client. *Committing peers* commits Txs and maintains the ledger. Note that a peer can play the role of endorser and committing peers at the same time.
3. Orderer: A node running the service of ordering Txs. Consensus algorithm is run between many orderers.
4. Auditor: The auditor can recover the sender identity, recipient identity and/or the Tx amount of any transaction.
5. Certificate Authority(CA): CA issues certificate for the public key of clients. Only authorized party can be involved in a Tx.

UTXO Model for Digital Assets. The model of Unspent Transaction Outputs (UTXO) is used in many blockchain systems, such as Bitcoin. If a user wants to spend his digital assets in a Tx, he has to refer to the specific assets that he wants to spend. If he spends the same asset twice, the verifier will notice it and will reject the Tx. In this paper, we consider the general UTXO model for digital assets and provides anonymity for their transactions[2].

Transaction Workflow. Assume that the client obtains a certificate from the CA. The basic workflow of a transaction (Tx) is as follows:

1. The client creates a signed Tx and sends it to endorser(s) of its choice.
2. Each endorser validates a Tx and produces an endorsement signature.
3. The submitting client collects endorsement(s) for a Tx and broadcasts it to the orderers.
4. The orderers deliver the block of ordered Txs to all peers.
5. The committing peer checks if every ordered Tx is endorsed correctly according to some policy. It also removes double-spending Tx endorsed by different endorsers concurrently (only the first valid Tx is accepted). If the checking is correct, it commits Txs and maintains the state and a copy of the ledger.

The auditor can recover the sender identity, recipient identity and the transaction amount of the Tx if necessary.

2.2 High-Level Description of PAChain

We briefly describe how we can achieve privacy and auditability in our PAChain. Generally speaking, we use the semi-trusted setting of consortium blockchain to set up system parameters and user credentials. Then, we can achieve a more efficient solutions for privacy and auditability, as compared to public blockchain (where all nodes may have Byzantine faults).

[2] Hyperledger Fabric 1.0 currently uses the account balance model by default, but it also supports the UTXO model.

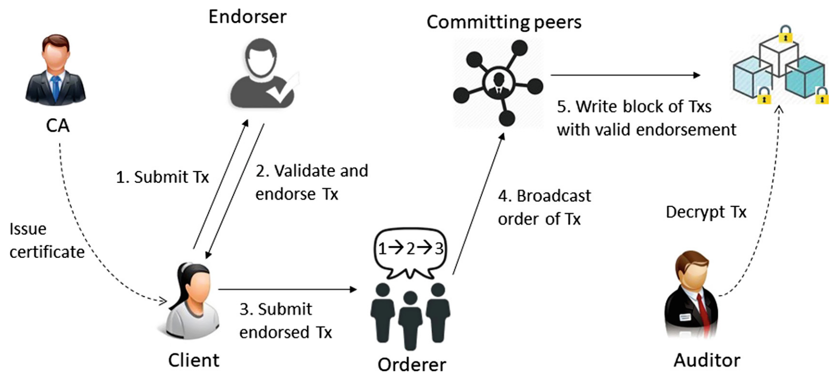

Fig. 1. Nodes and workflow of PAChain

Sender Privacy: It is achieved by the sender Alice using anonymous credential in Step 1 of the workflow. The credential is issued by the endorser of the last corresponding Tx received by Alice, to ensure that she is authenticated. We have to add a tag (which is a deterministic function of the user secret key) to the signature in Step 1 in order to detect double spending. We can take advantage of having some semi-trusted endorsers in our framework, which can act as the group manager of anonymous credential. It provides sender anonymity for all users using the same endorser (see Sect. 6.2). This anonymous credential approach is more efficient the ring signature-based approach in Monero and zk-SNARK-based approach in Zcash.

Recipient Privacy: It is achieved by generating one-time ephemeral key via Diffie-Hellman protocol between the sender and the recipient, in Step 1 of the workflow. However, we face the challenge that only authorized recipients are allowed in consortium blockchain. Therefore, we have to embed the zero-knowledge proof that the recipients are authorized into the generation of one-time ephemeral key (see Sect. 5.2). It is a new security requirement which does not exist in the public blockchain. This requirement is a major difference between PAChain and [8].

Transaction Privacy with Auditability: Our transaction amount privacy is achieved by using additive homomorphic encryption and zero-knowledge proof in Step 1 of the workflow. The proof shows that the Tx output amount is encrypted correctly, falls within a valid range, and the total Tx input and output amount are balanced. The major challenge is that using Paillier encryption with zero-knowledge range proof is not efficient. To be more specific, encrypting 2048-bit plaintext and performing the binary-decomposition range proof is an overkill for 64-bit of transaction amount (see Sect. 4). Therefore, we propose the use of modified ElGamal encryption with signature-based range proof to enhance the efficiency (see Sect. 4.2). The size of the zero knowledge proof is independent to the size of the range. It cannot be used in public blockchain since it requires trusted setup.

Auditability: Auditability for sender and recipient identity can be achieved by encrypting their public keys to the auditor, followed the zero-knowledge proof of the correctness of such encryption in Step 1 of the workflow. We have discussed the auditability of transaction amount above.

Our construction can also be modified to provide fine-grained privacy policy. For example, if the transaction amount M is under \$10,000, then the bank does not need to perform checking for anti-money laundering. As a result, we can leverage our efficient range proof to run the proof of knowledge:

$$\begin{aligned} PoK\{(M, &\text{sender ID, recipient ID}) : 0 \le M \le 10,000 \\ &\text{or Encrypt } (M, \text{sender ID, recipient ID}) \text{ to the auditor}\}. \end{aligned}$$

2.3 Threat Model

We assume that the system parameters are generated honestly. We consider the following attacker model for privacy:

- The attacker can create malicious client or corrupt any client.
- The peer and CA are assumed to be honest-but-curious: it tries to break privacy passively by recording all inputs, outputs and randomness used, but it still follows the protocols.
- All keys and data used by the orderers are known to the attacker.
- The auditor is assumed to be honest for privacy. Compromising the auditor trivially breaks all privacy[3].

We do not consider network-level privacy issues, such as tracing the sender's IP address or analyzing meta-data in network packets. The correctness of the consensus algorithm is not considered in this paper.

Consider the example of a consortium of banks. It is reasonable to assume that the banks jointly generate system parameters (e.g., by multi-party computation). Each bank acts as a peer and follows the protocol (if it ignores Tx or endorses invalid Tx, it will be discovered by other banks and will be handled by other means outside the blockchain system). However, the bank may be curious to view the Tx details of other banks. Our privacy model captures this scenario. As a result, we allow a larger degree of decentralization by allowing multiple endorsers to validate a Tx, without causing extra privacy leakage.

3 Backgrounds

3.1 Related Works

Public Blockchain. Monero is a cryptocurrency created in 2014 that provides privacy by linkable ring signature, stealth address and Ring Confidential

[3] One may argue that it gives too much power for auditor. However in most companies, internal auditor should always be able to control and governance business operations. In some industries, laws require that information must be provided to the court when requested (e.g., anti-money laundering in banks and lawful interception in telecommunication industry).

Transactions [11]. The major disadvantage of Monero is the size of the linkable ring signature, which is proportional to the size of the ring (related to the level of sender anonymity).

Zcash is a cryptocurrency created in 2016 that offers privacy and selective transparency of Txs by using zero-knowledge proofs (zk-SNARK) on special *shielded* transactions [2]. The major disadvantage of Zcash is the large signing key size of 896 MB, long key generation time of 8 min and long signing time of 3 min. As a result, only around 7% of Txs are shielded in Zcash as of March 2017. Recently, Zcash proposed a new Sapling update which reduced the proving time to a few seconds. However, it is still far less efficient than the Monero's approach (and also this paper's approach) which is about 100 ms.

A decentralized anonymous payment (DAP) with the support of accountability is proposed in [8]. They tackle the accountability problem in the public blockchain by using the zk-SNARK approach. Hence, it is also not efficient.

No authentication is provided for all solutions in the public blockchain.

Consortium Blockchain. For consortium blockchain, the major platforms provide limited support for privacy. In R3's Corda and Hyperledger Fabric, Txs are handled by different channels and users can only view Txs in their own channel. Therefore, privacy is maintained by the system's access control policy. The level of privacy is lower than that of Monero and Zcash, which are not affected by system administrators. In addition, Hyperledger Fabric uses *VKey* to provide transaction privacy by symmetric encryption. The problem of key distribution between all parties is a severe challenge for a global deployment of such system. There is an experiment to integrate identity mixer [7] with Fabric[4]. The identity mixer provides a better sender anonymity (by preventing the system administrator to link all transactions from the same user), and provides auditability for the sender identity.

Private Blockchain. Recently, a private industrial blockchain is proposed in [9], where multiple distributed private ledgers are maintained by a subset of stakeholders in the network. Each private ledger is maintained by its stakeholder only. This approach only provides privacy for users outside the private ledger. To provide higher level of privacy, the system must be divided into smaller private ledgers. However, more expensive cross ledger asset transfer is needed if there are some Txs between private ledgers.

PRCash [15] is a centrally-issued cryptocurrency with privacy and auditability. They provide anonymity of the sender and recipient identity. A mixing technique is used to obfuscate the relation between multiple inputs and outputs. However, there are still certain linkability between Txs. Anyone can see that the input of a Tx comes from which previous Tx output. For auditability, PRCash provides prefect transaction amount privacy and no auditability for small amount Tx. For Txs with large amount, it cannot provide privacy for these Txs.

[4] https://jira.hyperledger.org/browse/FAB-2005.

3.2 Mathematical Backgrounds

Bilinear Groups. $\mathcal{G}$ is an algorithm, which takes as input a security parameter λ and outputs a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups with prime order p, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a map, which has the following properties: (1) Bilinearity: $\hat{e}(g^a, \hat{g}^b) = \hat{e}(g, \hat{g})^{ab}$ for $\forall g \in \mathbb{G}_1, \hat{g} \in \mathbb{G}_2$ and $\forall a, b \in \mathbb{Z}_p$. (2) Non-degeneracy: There exists $g \in \mathbb{G}_1, \hat{g} \in \mathbb{G}_2$ such that $\hat{e}(g, \hat{g}) \neq 1_{\mathbb{G}}$. (3) Computability: There exists an efficient algorithm to compute $\hat{e}(g, \hat{g})$ for $\forall g \in \mathbb{G}_1, \hat{g} \in \mathbb{G}_2$.

4 Transaction Privacy

One of the challenging part for privacy in blockchain is the confidentiality of the transaction amount. The major difficulty is how to verify the Tx that (1) the total committed input amount is equal to the total committed output amount; (2) all committed amounts fall within a valid range, e.g., from 0 to 2^{64}. This requirement is commonly known as the *confidential transaction*. Theoretically, it can be achieved by combining additive homomorphic commitment with zero-knowledge range proof. One example is Monero [11], which uses Pedersen commitment with 1-out-of-2 ring signature-based binary decomposition range proof.

In PAChain, we require auditability of transaction amount. It is common in business use cases that all Txs can be audited by internal auditors or some external regulators. Therefore, it is necessary to replace additive homomorphic commitment with additive homomorphic encryption in confidential transaction, such that the decryption key is hold by the auditor. However, technical difficulties arise when combining the additive homomorphic Paillier encryption [12] with existing zero-knowledge range proof.

Range Proof with Encryption. Range proof is one essential element in transaction privacy. Without a range proof, the attacker can create a transaction with one input \$0 and two outputs of \$100 and −\$100. The sum of input and output amount is still balanced. If the output amount is encrypted, the attacker can create \$100 out of an input \$0. Therefore range proof is needed to prevent any negative amount or overflow amount.

Problems of Using Paillier Encryption with Range Proof. Paillier encryption [12] is the most common additive homomorphic encryption to date. However, combining Paillier encryption with existing range proof is not trivial in blockchain applications. Therefore, there is no simple and efficient solution for using Paillier encryption with range proof in blockchain.

4.1 Security Model of Transaction Privacy Protocol

We give a high level description on the notion and security model for transaction privacy. Details will be given in the full version of the paper.

A transaction privacy protocol consists of:

- Setup: It outputs the system parameters and the secret key of an auditor.
- TxPrivacySpend: When given some UTXO input amount, UTXO input ciphertexts and some output amount, it outputs some UTXO output ciphertexts and the proof of correctness.
- TxPrivacyVerify: When given some UTXO input ciphertexts, some UTXO output ciphertexts and the proof of correctness, it checks if the proof is correct or not.
- Decrypt: When given a UTXO ciphertext and the secret key of an auditor, it outputs the decrypt amount.

Security Model. We define the security requirements for transaction privacy:

- No output transaction amount is outside the range with a valid proof in TxPrivacySpend.
- The total input transaction amount is equal to the total output transaction amount in TxPrivacySpend.
- No one can learn the transaction amount from the ciphertext, except the auditor.

4.2 Transaction Privacy for PAChain

We give our efficient transaction privacy solution for consortium blockchain. We overcome the problem for effectively combining additive homomorphic encryption and range proof due to two properties: (1) consortium blockchain allows trusted setup; (2) the transaction amount is short.

The first observation is that encrypting a "short" transaction amount (e.g., 51-bit can represent all 21M Bitcoins in terms of satoshi, or 64-bit can represent trillions of dollars with sixth decimals) with Paillier encryption of message space of 2048-bit is superfluous. Therefore, we propose to use the additive homomorphic ElGamal encryption instead. However, decrypting such ciphertext requires the computation of discrete logarithm, which is not feasible for large message space. Hence, we decompose the K-bit transaction amount M into ℓ segment $\mu_0, \dots, \mu_{\ell-1}$ which are smaller than the message space $\mathfrak{u}$ by $M = \sum_{j=0}^{\ell-1} \mu_j \mathfrak{u}^j$. As a result, we encrypt each μ_j by additive homomorphic ElGamal encryption. The small message space of $\mathfrak{u}$ guarantees efficient decryption. In our implementation, we consider 64-bit transaction amount and the auditor uses a pre-computation table of $(g, g^2, \dots, g^{\mathfrak{u}-1})$ for efficient decryption. We can choose $\ell = 4, \mathfrak{u} = 65536$ and the pre-computation table is about 2 MB. The ciphertext size is 2048-bit, which is still less than the 4096-bit ciphertext size of Paillier encryption.

Another advantage of using ElGamal encryption is the ease to combine with range proof. By using ECC ElGamal encryption, it can be combined with the Boneh-Boyen signature-based range proof [6]. Note that this solution is only suitable for consortium blockchain since the non-interactive version of such range proof requires a trusted setup.

Our Construction. Our transaction privacy (TP) protocol is described below.

- Setup. On input a security parameter 1^λ and the range parameter $R = \mathfrak{u}^\ell$, it generates the bilinear group by $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. It randomly picks generators $g, g_0 \in \mathbb{G}_1$, $\hat{g} \in \mathbb{G}_2$ and $\mathfrak{X} \in \mathbb{Z}_p$. It computes $\hat{\mathfrak{Y}} = \hat{g}^{\mathfrak{X}}$, $\mathfrak{A}_i = g^{\frac{1}{\mathfrak{X}+i}}$ for $i \in [0, \mathfrak{u}-1]$. Suppose $H : \{0,1\}^* \to \mathbb{Z}_p$ is a collision resistant hash function. In addition, suppose the auditor picks a random secret key $\mathsf{ask_{tp}}$ in $\mathbb{Z}_p$ and outputs its public key $h_{\mathsf{tp}} = g^{\mathsf{ask_{tp}}}$. It outputs $\mathsf{param} = (g, g_0, h_{\mathsf{tp}}, \hat{g}, \hat{e}(g, \hat{g}), \hat{\mathfrak{Y}}, \mathfrak{u}, \ell, H, \mathfrak{A}_0, \ldots, \mathfrak{A}_{\mathfrak{u}-1})$.

- $\mathsf{TxPrivacySpend}$. On input n' output transaction amount $M_{\mathsf{out},j}$, n transaction input ciphertext $C_{\mathsf{in},i}$, amount $M_{\mathsf{in},i}$ and randomness $r_{\mathsf{in},i}$, it outputs $\perp$ if $C_{\mathsf{in},i}$ is not a valid ciphertext for $(M_{\mathsf{in},i}, r_{\mathsf{in},i})$ for some $i \in [1, n]$, or $\sum_{i=1}^{n} M_{\mathsf{in},i} \neq \sum_{j=1}^{n'} M_{\mathsf{out},j}$.

For each $M_{\mathsf{out},j}$, it obtains $(C_{\mathsf{out},j}, r_{\mathsf{out},j}, \pi_{\mathsf{enc},j})$ by running the sub-protocol $\mathsf{EncProof}(M_{\mathsf{out},j})$.

$\mathsf{EncProof}$. Suppose that the prover wants to prove some M lies in $[0, \mathfrak{u}^\ell)$. It runs as follows:

1. It first decomposes M into $\mu_k \in [0, \mathfrak{u}-1]$ such that $M = \sum_{k=0}^{\ell-1} \mu_k \mathfrak{u}^k$.
2. For each μ_k, the prover computes the ElGamal ciphertext $(C_k = g_0^{\mu_k} h_{\mathsf{tp}}^{r_k}$, $B_k = g^{r_k})$ for some random $r_k \in \mathbb{Z}_p$. Denote $C_{\mathsf{enc}} = \{C_k,\ B_k\}_{k \in [0,\ell-1]}$ and $r_{\mathsf{enc}} = \sum_{k=0}^{\ell-1} r_k \mathfrak{u}^k$.
3. For each μ_k, it proves in zero-knowledge that the encrypted μ_k corresponds to some $\mathfrak{A}_{\mu_k}$:
$$\pi_{\mathsf{enc}} \leftarrow PoK\{(\{\mu_k, r_k, \mathfrak{A}_{\mu_k}\}_{k\in[0,\ell-1]}) : \hat{e}(g, \hat{g}) = \hat{e}(\mathfrak{A}_{\mu_k}, \hat{g}^{\mu_k} \cdot \hat{\mathfrak{Y}}) \wedge C_k = g_0^{\mu_k} h_{\mathsf{tp}}^{r_k} \wedge B_k = g^{r_k}\}.$$
Details of the zero-knowledge proof is as follows. The prover randomly picks $v_k, s_k, t_k, \nu \in \mathbb{Z}_p$ for $k \in [0, \ell-1]$ and computes:
$$V_k = \mathfrak{A}_{\mu_k}^{v_k}, \quad a_k = \hat{e}(V_k, \hat{g})^{-s_k} \cdot \hat{e}(g, \hat{g})^{t_k}, \quad E_k = g_0^{s_k} h_{\mathsf{tp}}^{\nu_k}, \quad D_k = g^{\nu_k}.$$
It computes $\tilde{c} = H(\mathsf{param}, \{V_k, a_k, B_k, C_k, D_k,\ E_k\}_{k\in[0,\ell-1]})$ and for $k \in [0, \ell-1]$:
$$z_{\mu_k} = s_k - \tilde{c}\mu_k, \quad z_{v_k} = t_k - \tilde{c}v_k, \quad z_{r_k} = \nu_k - \tilde{c}r_k.$$
Then $\pi_{\mathsf{enc}} = (\{V_k, z_{\mu_k},\ z_{v_k}, z_{r_k}\}_{k\in[0,\ell-1]}, \tilde{c})$.
4. It outputs $(C_{\mathsf{enc}}, r_{\mathsf{enc}}, \pi_{\mathsf{enc}})$.

Observe that $C_{\mathsf{out},j} = \{C_{j,k}, B_{j,k}\}_{k\in[0,\ell-1]}$. For simplicity, denote $C'_{\mathsf{out},j} = \prod_{k=0}^{\ell-1} C_{j,k}^{\mathfrak{u}^k}$ and $B'_{\mathsf{out},j} = \prod_{k=0}^{\ell-1} B_{j,k}^{\mathfrak{u}^k}$. The same definition applies for input ciphertext.

Next, it proves that the total input Tx amount is equal to the total output Tx amount. It is equivalent to know $x_{\mathsf{tp}} = \sum_{j=1}^{n'} r_{\mathsf{out},j} - \sum_{i=1}^{n} r_{\mathsf{in},i}$, such that

$$\prod_{j=1}^{n'} C'_{\mathsf{out},j} / \prod_{i=1}^{n} C'_{\mathsf{in},i} = h_{\mathsf{tp}}^{x_{\mathsf{tp}}}.$$

The zero knowledge proof of x_{tp} is as follows. It picks some random $r_{\mathsf{tp}} \in \mathbb{Z}_p$ and computes $R_{\mathsf{tp}} = h_{\mathsf{tp}}^{r_{\mathsf{tp}}}$, $\tilde{R}_{\mathsf{tp}} = g^{r_{\mathsf{tp}}}$, $\tilde{c}' = H(\mathsf{param}, R_{\mathsf{tp}}, \tilde{R}_{\mathsf{tp}}, \{C_{\mathsf{in},i}\}_{i\in[1,n]}, \{C_{\mathsf{out},j}, \pi_{\mathsf{enc},j}\}_{j\in[1,n']})$, $z_{\mathsf{tp}} = r_{\mathsf{tp}} + \tilde{c}' x_{\mathsf{tp}}$. Denote $\pi_{\mathsf{tp}} = (z_{\mathsf{tp}}, \tilde{c}', \{\pi_{\mathsf{enc},j}\}_{j\in[1,n']})$.

The algorithm outputs $(\{C_{\mathsf{out},j}, r_{\mathsf{out},i}\}_{j\in[1,n']}, \pi_{\mathsf{tp}})$.

- TxPrivacyVerify. On input n Tx input ciphertext $C_{\mathsf{in},i}$, n' Tx output ciphertext $C_{\mathsf{out},j}$ and a proof $\pi_{\mathsf{tp}} = (z_{\mathsf{tp}}, \tilde{c}', \{\pi_{\mathsf{enc},j}\}_{j\in[1,n']})$. For each $\pi_{\mathsf{enc},j}$, it runs the following sub-protocol:

EncVerify. On input the ciphertext $C_{\mathsf{out},j} = \{C_k, B_k\}_{k\in[0,\ell-1]}$ and the proof $\pi_{\mathsf{enc},j} = (\{V_k, z_{\mu_k}, z_{v_k}, z_{r_k}\}_{k\in[0,\ell-1]}, \tilde{c})$, it validates the proof by computing for all $k \in [0, \ell-1]$:

$$D_k = B_k^{\tilde{c}} g^{z_{r_k}}, \quad E_k = C_k^{\tilde{c}} g_0^{z_{\mu_k}} h_{\mathsf{tp}}^{z_{r_k}}, \quad a_k = \hat{e}(V_k, \hat{\mathfrak{Y}}^{\tilde{c}} \hat{g}^{-z_{\mu_k}}) \cdot \hat{e}(g, \hat{g})^{z_{v_k}}.$$

It outputs 1 if $\tilde{c} = H(\mathsf{param}, \{V_k, a_k, B_k, C_k, D_k, E_k\}_{k\in[0,\ell-1]})$ or 0 otherwise.

It computes $R'_{\mathsf{tp}} = h_{\mathsf{tp}}^{z_{\mathsf{tp}}} (\frac{\prod_{i=1}^{n} C'_{\mathsf{in},i}}{\prod_{j=1}^{n'} C'_{\mathsf{out},j}})^{\tilde{c}'}$, $\tilde{R}'_{\mathsf{tp}} = g^{z_{\mathsf{tp}}} (\frac{\prod_{i=1}^{n} B'_{\mathsf{in},i}}{\prod_{j=1}^{n'} B'_{\mathsf{out},j}})^{\tilde{c}'}$. It returns 1 if and only if all EncVerify outputs 1, and $\tilde{c}' = H(\mathsf{param}, R'_{\mathsf{tp}}, \tilde{R}'_{\mathsf{tp}}, \{C_{\mathsf{in},i}\}_{i\in[1,n]}, \{C_{\mathsf{out},j}, \pi_{\mathsf{enc},j}\}_{j\in[1,n']})$.

- Decrypt. On input the auditor's secret key $\mathsf{ask}_{\mathsf{tp}}$ and a ciphertext $C_{\mathsf{enc}} = \{C_k, B_k\}_{k\in[0,\ell-1]}$, it computes $g_0^{\mu_k} = \frac{C_k}{B_k^{\mathsf{ask}_{\mathsf{tp}}}}$ for $k \in [0, \ell-1]$. The auditor uses a pre-computation table containing $(g_0^0, g_0^1, \ldots, g_0^{\mathfrak{u}-1})$ to find out the value of μ_k. Finally, the auditor recovers $M = \sum_{k=0}^{\ell-1} \mu_k \mathfrak{u}^k$.

Security of Transaction Privacy. We give the security theorem of our TP protocol. The proofs are given in the full version of the paper.

Theorem 1. *Our TP protocol is sound if the $\mathfrak{u}$-Strong Diffie-Hellman (SDH) assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ in the random oracle model. Our TP protocol is private if the decisional Diffie-Hellman (DDH) assumption holds in $\mathbb{G}_1$ in the random oracle model. Our TP protocol is balance if the discrete logarithm (DL) assumption holds in $\mathbb{G}_1$ in the random oracle model.*

5 Recipient Privacy

In blockchain, the user address is the hash of his public key, and hence it represents his identity. If we want to preserve the recipient privacy, we can always use a new public key for each Tx. However, this approach is problematic in some consortium blockchain which only allows Txs between authenticated users. It means that all recipient (and sender) address should be authenticated. A straightforward approach is to associate each address with a certificate issued by a CA. The key challenge is how to validate the certificate while hiding the public key/address at the same time.

Previous Works. Dash's PrivateSend is a coin-mixing service based on CoinJoin [13]. Dash requires combining identical input amounts from multiple senders at the time of mixing, and thus it restricts the mixing to only accept certain denominations (e.g. \$0.1, \$1, \$10, etc.). The level of anonymity is related to number of Txs mixed. In Monero, the recipient uses *stealth address* [11], which is a one-time DH-type public key computed from the recipient's public key and some randomness included in the transaction block. The corresponding one-time secret key is only computable by the recipient. In Zcash, it uses the general zk-SNARK to provide zero knowledge for all Tx details including UTXO used [2].

5.1 Security Model of Recipient Privacy Protocol

The formal security notion and security models will be given to the full version of the paper. Roughly speaking, the security requirements for recipient privacy are *soundness* and *anonymity*. It includes:

1. No adversary can be a recipient without credential, even with colluding auditor.
2. No one can learn the identity of the recipient, except the auditor.

5.2 Recipient Privacy for PAChain

Stealth address [11] appears to be the most efficient approach for recipient privacy. However in consortium blockchain, only the recipient's public key is authenticated by the CA, but not the one-time public key. Therefore, the sender additionally needs to show that the one-time public key is computed from an authenticated public key, without revealing the public key itself.

Our Construction. The recipient's certificate is signed by the CA using BBS+ signature [1], which allows efficient zero-knowledge proof. In addition, we encrypt the long-term public key in the zero-knowledge proof, such that the auditor can decrypt the real address (long-term public key) of the recipient.

Our recipient privacy (RP) protocol is described below.

– Setup. On input a security parameter 1^λ, the setup algorithm generates the bilinear group by $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. It picks some random generators $g, g_2, g_3, h_2 \in \mathbb{G}_1$ and $\hat{g}_2 \in \mathbb{G}_2$. Suppose $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H' : \mathbb{G}_1 \rightarrow \mathbb{Z}_p$ are collision resistant hash functions. In addition, suppose the auditor picks a random secret key $\mathsf{ask}_{\mathsf{rp}} \in \mathbb{Z}_p$ and outputs its public key $h_{\mathsf{rp}} = g^{\mathsf{ask}_{\mathsf{rp}}}$. It outputs the public parameters $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g, h_{\mathsf{rp}}, g_2, g_3, h_2, \hat{g}_2, H, H')$.

- $\mathsf{UserKeyGen}$. The user randomly picks a long-term secret key $x_1, x_2 \in \mathbb{Z}_p$ and computes a long-term public key $Y_1 = g_2^{x_1}, Y_2 = g_2^{x_2}$. It outputs the user key pair $(\mathsf{usk} = (x_1, x_2), \mathsf{upk} = (Y_1, Y_2))$.
- $\mathsf{OneTimePkGen}$. On input $\mathsf{upk} = (Y_1, Y_2)$, the sender randomly picks $r_{tx} \in \mathbb{Z}_p$ and outputs $(R_{tx} = g_2^{r_{tx}}, \mathsf{otpk} = Y_1 g_2^{H'(Y_2^{r_{tx}})})$.
- $\mathsf{OneTimeSkGen}$. On input otpk, R_{tx} and $\mathsf{usk} = (x_1, x_2)$, the recipient computes a one-time secret key $\mathsf{otsk} = x_1 + H'(R_{tx}^{x_2})$, and it outputs otsk if $\mathsf{otpk} = g_2^{\mathsf{otsk}}$.
- $\mathsf{CAKeyGen}$. The CA randomly picks $\beta \in \mathbb{Z}_p$ and computes $\hat{W}_2 = \hat{g}_2^{\beta}$. It outputs the CA key pair $(\mathsf{cask} = \beta, \mathsf{capk} = \hat{W}_2)$.
- $\mathsf{CertIssue}$. On input CA's public key capk and the user long-term public key $\mathsf{upk} = (Y_1, \cdot)$, the user first performs a zero-knowledge proof of discrete logarithm: $x_1 = \log_{g_2} Y_1$. Denote this proof as π_{ca}. After the CA validates π_{ca}, the CA picks some random $s, w \in \mathbb{Z}_p$ and uses his private key $\mathsf{cask} = \beta$ to compute: $F = (h_2 \cdot Y_1 \cdot g_3^s)^{\frac{1}{\beta + w}}$. The CA returns the certificate (F, w, s) to the user.
- $\mathsf{RecPrivacySpend}$. On input param, $\mathsf{capk} = \hat{W}_2$:

1. The sender with usk decides one or more UTXOs that he wants to spend. For simplicity, assume he picks one UTXO with $(\mathsf{otpk}_s, R_{tx,s})$. He runs $\mathsf{otsk}_s \leftarrow$ **OneTimeSkGen**$(\mathsf{param}, \mathsf{otpk}_s, R_{tx,s}, \mathsf{usk})$. It runs a zero-knowledge proof of discrete logarithm: $\mathsf{otsk}_s = \log_{g_2} \mathsf{otpk}_s$. Denote this proof as π_{otsk}.
2. The sender chooses the set of recipients. For simplicity, assume there is only one recipient with long-term public key $\mathsf{upk}_r = (Y_1, Y_2)$. The sender generates the recipient's one-time public key by running **OneTimePkGen**. The sender obtains $\mathsf{otpk}_r, R_{tx,r}$ and the randomness r_{tx}. Denote $h_{tx} = H'(Y_2^{r_{tx}})$.
3. The sender encrypts Y_1 to the auditor by picking a random $r_{\mathsf{cert}} \in \mathbb{Z}_p$ and computing $C_{\mathsf{rp}} = (C_{\mathsf{cert}} = Y_1 \cdot h_{\mathsf{rp}}^{r_{\mathsf{cert}}}, B_{\mathsf{cert}} = g^{r_{\mathsf{cert}}})$.
4. The sender runs the following proof of knowledge for showing that (1) otpk_r is computed from a public key, (2) the public key has a valid certificate (F, w, s), (3) the public key is encrypted to the auditor:

$$\pi_{\mathsf{rp}} \leftarrow PoK\{(F, w, s, Y_1, h_{tx}, r_{\mathsf{cert}}) : \hat{e}(F, \hat{g}_2^w \cdot \hat{W}_2) = \hat{e}(h_2 \cdot Y_1 \cdot g_3^s, \hat{g}_2) \\ \wedge\ \mathsf{otpk}_r = Y_1 g_2^{h_{tx}} \wedge B_{\mathsf{cert}} = g^{r_{\mathsf{cert}}} \wedge C_{\mathsf{cert}} = Y_1 h_{\mathsf{rp}}^{r_{\mathsf{cert}}}\}.$$

The details of the zero knowledge proof π_{rp} is as follows.

(a) **ZKCommit:** It picks some random $\rho, r_\tau, r_\omega, r_\sigma, r_\rho, r_{\mathsf{cert}}, r_c, r_s \in \mathbb{Z}_p$, computes $\Theta = F^\rho$ and

$$R_{\mathsf{cert},1} = \hat{e}((h_2 \cdot C_{\mathsf{cert}})^{r_\rho} g_3^{r_s} \Theta^{-r_\omega} h_{\mathsf{rp}}^{-r_\sigma}, \hat{g}_2),$$
$$R_{\mathsf{cert},2} = g^{r_c},\ R_{\mathsf{cert},3} = B_{\mathsf{cert}}^{r_\rho} g^{-r_\sigma},\ R_{\mathsf{cert},4} = h_{\mathsf{rp}}^{r_c} g_2^{-r_\tau}.$$

(b) **ZKChallenge:** It computes $c = H(\mathsf{CertAuth.mpk}, C_{\mathsf{rp}}, \Theta, R_{\mathsf{cert},1}, R_{\mathsf{cert},2}, R_{\mathsf{cert},3}, R_{\mathsf{cert},4})$.

(c) **ZKResponse:** It computes:

$$z_\omega = r_\omega + c \cdot w, \qquad z_\tau = r_\tau + c \cdot h_{tx}, \qquad z_\rho = r_\rho + c \cdot \rho,$$
$$z_c = r_c + c \cdot r_{\mathsf{cert}}, \qquad z_\sigma = r_\sigma + c \cdot r_{\mathsf{cert}} \cdot \rho, \qquad z_s = r_s + c \cdot \rho \cdot s.$$

It outputs $\pi_{\mathsf{rp}} = (c, \Theta, z_\omega, z_\tau, z_\rho, z_c, z_\sigma, z_s)$.

5. Output $\pi_{\mathsf{otsk}}, \mathsf{otpk}_s, \mathsf{otpk}_r, R_{tx,r}, C_{\mathsf{rp}}$ and π_{rp}.

- RecPrivacyVerify. On input param, $\mathsf{capk}, \pi_{\mathsf{otsk}}, \mathsf{otpk}_s, \mathsf{otpk}_r$, $R_{tx,r}$, C_{rp} and π_{rp}, it outputs 1 if π_{otsk} and π_{rp} are valid zero knowledge proofs.

The details of verifying the zero knowledge proof $\pi_{\mathsf{rp}} = (c, \Theta, z_\omega, z_\tau, z_\rho, z_c, z_\sigma, z_s)$ is as follows.

1. **ZKReconstruct:** Denote $C_{\mathsf{rp}} = (C_{\mathsf{cert}}, B_{\mathsf{cert}})$. It computes:

$$R_{\mathsf{cert},1} = \hat{e}((h_2 \cdot C_{\mathsf{cert}})^{z_\rho} g_3^{z_s} \Theta^{-z_\omega} h_{\mathsf{rp}}^{-z_\sigma}, \hat{g}_2) \cdot \hat{e}(\Theta, \hat{W}_2)^c,$$
$$R_{\mathsf{cert},2} = g^{z_c} B_{\mathsf{cert}}^{-c}, \quad R_{\mathsf{cert},3} = B_{\mathsf{cert}}^{z_\rho} g^{-z_\sigma}, \quad R_{\mathsf{cert},4} = h_{\mathsf{rp}}^{z_c} g_2^{-z_\tau} (\mathsf{otpk}_r / C_{\mathsf{cert}})^c.$$

2. **ZKCheck:** It computes $c' = H(\mathsf{CertAuth.mpk}, C_{\mathsf{rp}}, \Theta, R_{\mathsf{cert},1}, R_{\mathsf{cert},2}, R_{\mathsf{cert},3}, R_{\mathsf{cert},4})$. If $c = c'$, then π_{rp} is a valid zero knowledge proof.

Security of Recipient Privacy. We give the security theorem of our RP protocol. The proofs are given in the full version of the paper.

Theorem 2. *Our RP protocol is sound if the q-SDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *in the random oracle model, where q is the maximum number of* Issue *oracle query. Our RP protocol is anonymous if the DDH assumption holds in* $\mathbb{G}_1$ *in the random oracle model.*

6 Sender Privacy

In the UTXO model, the sender has to specify the UTXOs that he wants to use. The UTXOs include the information of the owner's address as well as the transaction amount. The linkage between the current transaction and UTXOs guarantees the validity of the transaction and ensures that there is no double spending. However, this linkage violates the privacy of the sender (no matter the address is used for one time only and the transaction amount is encrypted). It is a dilemma to preserve the transaction correctness and to protect the sender privacy at the same time.

Previous Works. The sender privacy for Dash and Zcash are achieved as the same way as the recipient privacy. In Monero, it uses linkable ring signature (LRS) for hiding the real UTXOs used with other UTXOs (by the anonymity property of LRS), preventing double spending (by the linkability property of LRS) and ensuring transaction correctness (by the unforgeability property of LRS) at the same time [11]. The level of anonymity is related to number of UTXOs (denote as L) included in LRS. However, the number of computation used in signing and the signature size are both $O(L)$. Recently, Sun *et al.* reduced the signature size to $O(1)$ [14], at the price of using trusted setup.

6.1 Security Model of Sender Privacy Protocol

The formal security notion and model for the sender privacy protocol will be given in the full version of the paper. In short, the security requirements for sender privacy are *soundness, unforgeability* and *anonymity*. It includes:

1. No adversary can spend without credential, even with colluding auditor.
2. No adversary can spend money of honest user, even with colluding endorser and auditor.
3. No one can learn the identity of the sender, except the auditor.

6.2 Sender Privacy for PAChain

We give our efficient sender privacy solution for consortium blockchain. By the semi-trusted property of consortium blockchain, we can use the anonymous credential approach to achieve sender privacy. By using the semi-trusted endorser as the group manager (in the honest-but-curious security model), we provide an efficient solution which has the signing time, verification time and signature size independent to the number of UTXO included in the group. At the same time, the sender can be revealed by the auditor. Note that similar to group signature, the endorser (who issued credentials) cannot link the transaction by the credential he issued. Credential is issued to the recipient when the endorser approve the transaction. The endorser does not have any advantage in breaking anonymity in the UTXO model.

Our Construction. Our construction differs from traditional group signatures in two ways: (1) we have to hide both the sender's public key as well as the Tx amount, (2) we have to add a linkability tag to avoid double spending. For the first requirement, we use the BBS group signature [4], since the underlying credential is signed by Boneh-Boyen signature [3], which can be modified to sign on multiple committed values [1]. For the second requirement, we use the tag structure used in most linkable ring signature schemes.

There are two possible constructions: the Tx amount is in plaintext or in ciphertext. In the UTXO model, transaction privacy is required in order to protect sender privacy (otherwise, the attacker can use the Tx amount to link past transactions). In the account-based model, transaction privacy may or may not be needed in the blockchain. For simplicity, we only give the Tx amount ciphertext version here and the plaintext version can be constructed similarly.

– Setup. On input a security parameter 1^λ, the setup algorithm generates the bilinear group by $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$. It picks some random generators $g, g_1, g_2, u_1, h_s, f \in \mathbb{G}_1$ and $\hat{g}_2 \in \mathbb{G}_2$. Suppose $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ is a collision resistant hash function. Denote h_{tp} as the public key of the auditor in transaction privacy. Suppose the auditor picks a random secret key $\mathsf{ask}_{\mathsf{sp}}$ in $\mathbb{Z}_p$ and outputs its public key $h_{\mathsf{sp}} = g^{\mathsf{ask}_{\mathsf{sp}}}$. It outputs the public parameters $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g, g_1, g_2, u_1, h_s, f, h_{\mathsf{tp}}, h_{\mathsf{sp}}, \hat{g}_2, H)$.

- $\mathsf{UserKeyGen}$. The user secret key is $\mathsf{usk} = x_1 \in \mathbb{Z}_p$ and the public key is $\mathsf{upk} = Y_1 = g_2^{x_1}$[5].
- $\mathsf{EndorserKeyGen}$. The endorser randomly picks $\alpha \in \mathbb{Z}_p$ and computes $W_{\mathsf{sp}} = \hat{g}_2^{\alpha}$. It outputs the endorser key pair $(\mathsf{esk} = \alpha, \mathsf{epk} = W_{\mathsf{sp}})$.
- $\mathsf{CredIssue}$. On input endorser public key epk, the user public key Y_1 and the Tx amount ciphertext C, the user first performs a zero-knowledge proof of secret key: $x_1 = \log_{g_2} Y_1$. Denote this proof as π_{ci}. The user sends π_{ci} and π_{tp} to the endorser, where π_{tp} is the zero-knowledge transaction privacy proof (showing the knowledge of (m, r_{tp}) such that $C = g^m h_{\mathsf{tp}}^{r_{\mathsf{tp}}}$).
  After the endorser validates the proofs π_{ci} and π_{tp}, the endorser picks some random $v, z \in \mathbb{Z}_p$ and uses his secret key $\mathsf{esk} = \alpha$ to compute: $A = (h_s \cdot g_1^v \cdot C \cdot Y_1)^{\frac{1}{\alpha+z}}$. The endorser returns the credential $\mathsf{cred} = (A, v, z)$ to the user.
- $\mathsf{CredSign}$. On input param, and private input tuples $x'_{\mathsf{in}}, \mathsf{cred}_{\mathsf{in}}$, $m'_{\mathsf{in}}, r'_{\mathsf{in}}, Y'_{\mathsf{in}}$ (such that $C_{\mathsf{in}} = g^{m'_{\mathsf{in}}} h_{\mathsf{tp}}^{r'_{\mathsf{in}}}$), it runs the following:

1. It computes the tag for detecting double spending: $T = f^{x'_{\mathsf{in}}}$.
2. It encrypts the public key Y'_{in} to the auditor, by randomly choosing $r_{\mathsf{cred}} \in \mathbb{Z}_p$ and computing $C_{\mathsf{sp}} = (C_{\mathsf{cred}} = Y'_{\mathsf{in}} \cdot h_{\mathsf{sp}}^{r_{\mathsf{cred}}}, B_{\mathsf{cred}} = g^{r_{\mathsf{cred}}})$.
3. It computes the zero knowledge proof π_{sp} for: (1) the credential $\mathsf{cred}_{\mathsf{in}} = (A, v, z)$ corresponds to $Y'_{\mathsf{in}} = g_2^{x'_{\mathsf{in}}}$ and $C_{\mathsf{in}} = g^{m'_{\mathsf{in}}} h_{\mathsf{tp}}^{r'_{\mathsf{in}}}$; (2) $T = f^{x'_{\mathsf{in}}}$; (3) Y'_{in} is encrypted to the auditor.

$$\begin{aligned}\pi_{\mathsf{sp}} =& PoK\{(x'_{\mathsf{in}}, m'_{\mathsf{in}}, r'_{\mathsf{in}}, A, v, z, r_{\mathsf{cred}}) : \hat{e}(A, W_{\mathsf{sp}}\hat{g}_2^z) = \hat{e}(h_s g_1^v g^{m'_{\mathsf{in}}} h_{\mathsf{tp}}^{r'_{\mathsf{in}}} g_2^{x'_{\mathsf{in}}}, \hat{g}_2)\\ &\wedge T = f^{x'_{\mathsf{in}}} \wedge C_{\mathsf{cred}} = g_2^{x'_{\mathsf{in}}} \cdot h_{\mathsf{sp}}^{r_{\mathsf{cred}}} \wedge B_{\mathsf{cred}} = g^{r_{\mathsf{cred}}}\}.\end{aligned}$$

The output signature $\sigma = (\pi_{\mathsf{sp}}, C_{\mathsf{sp}}, T)$. Details of the zero-knowledge proof is shown as follows.

(a) **ZKCommit:** It picks some random $a, r_\psi, r_k, r_a, r_b, r_z, r_m, r_r, r_v \in \mathbb{Z}_p$. It computes:

$$\begin{aligned}&S = A \cdot u_1^a, \quad \varXi = g_1^a,\\ &R_{\mathsf{cred},1} = \hat{e}(u_1^{r_b} S^{-r_z} g_1^{r_v} g^{r_m} h_{\mathsf{tp}}^{r_r} g_2^{r_k}, \hat{g}_2) \cdot \hat{e}(u_1, W_{\mathsf{sp}})^{r_a}, \; R_{\mathsf{cred},2} = g_1^{r_a},\\ &R_{\mathsf{cred},3} = \varXi^{r_z} g_1^{-r_b}, \; R_{\mathsf{cred},4} = g^{r_\psi}, \; R_{\mathsf{cred},5} = g_2^{r_k} h_{\mathsf{sp}}^{r_\psi}, \; R_{\mathsf{cred},6} = f^{r_k}.\end{aligned}$$

(b) **ZKChallenge:** It computes $c = H(\mathsf{CredAuth.mpk}, C_{\mathsf{sp}}, T, S, \varXi, R_{\mathsf{cred},1}, R_{\mathsf{cred},2}, R_{\mathsf{cred},3}, R_{\mathsf{cred},4}, R_{\mathsf{cred},5}, R_{\mathsf{cred},6})$.
(c) **ZKResponse:** It computes:

$$\begin{aligned}&z_k = r_k + c \cdot x'_{\mathsf{in}}, && z_a = r_a + c \cdot a, && z_z = r_z + c \cdot z,\\ &z_b = r_b + c \cdot a \cdot z, && z_v = r_v + c \cdot v, && z_m = r_m + c \cdot m'_{\mathsf{in}},\\ &z_r = r_r + c \cdot r'_{\mathsf{in}}, && z_\psi = r_\psi + c \cdot r_{\mathsf{cred}}.\end{aligned}$$

It outputs the proof $\pi_{\mathsf{sp}} = (c, S, \varXi, z_k, z_a, z_z, z_b, z_v, z_m, z_r, z_\psi)$.

[5] This public key Y_1 can be a long term public key if recipient anonymity is not protected in the previous transaction. Otherwise, it can be a one-time public key.

- Verify. On input param, the endorser public keys W_s, a signature $\sigma = (\pi_{\mathsf{sp}}, C_{\mathsf{sp}} = (C_{\mathsf{cred}}, B_{\mathsf{cred}}), T)$, it checks the validity of the proof $\pi_{\mathsf{sp}} = (c, S, \Xi, z_k, z_a, z_z, z_b, z_v, z_m, z_r, z_\psi)$:

1. **ZKReconstruct:** It computes:

$$\begin{aligned} R'_{\mathsf{cred},1} &= \hat{e}(u_1^{z_b} S^{-z_z} g_1^{z_v} g^{z_m} h_{\mathsf{tp}}^{z_r} g_1^{z_k} h_s^c, \hat{g}_2) \cdot \hat{e}(u_1^{z_a} S^{-c}, W_{\mathsf{sp}}), \\ R'_{\mathsf{cred},2} &= g_1^{z_a} \Xi^{-c}, \quad R'_{\mathsf{cred},3} = \Xi^{z_z} g_1^{-z_b}, \quad R'_{\mathsf{cred},4} = g^{z_\psi} B_{\mathsf{cred}}^{-c}, \\ R'_{\mathsf{cred},5} &= g_2^{z_k} h_{\mathsf{sp}}^{z_\psi} C_{\mathsf{cred}}^{-c}, \quad R'_{\mathsf{cred},6} = f^{z_k} T^{-c}. \end{aligned}$$

2. **ZKCheck:** It computes $c' = H(\mathsf{CredAuth.mpk}, C_{\mathsf{sp}}, T, S, \Xi, R_{\mathsf{cred},1}, R_{\mathsf{cred},2}, R_{\mathsf{cred},3}, R_{\mathsf{cred},4}, R_{\mathsf{cred},5}, R_{\mathsf{cred},6})$.

It outputs 1 if $c = c'$; and outputs 0 otherwise.

- Link. On input param and two tags T_1, T_2 in signatures σ_1, σ_2, such that $T_1 = T_2$, it outputs 1. Otherwise it outputs 0.

- $\mathsf{Decrypt}$. On input a ciphertext $(C_{\mathsf{cred}}, B_{\mathsf{cred}})$ and $\mathsf{ask}_{\mathsf{sp}}$, it computes $Y' = C_{\mathsf{cred}} / B_{\mathsf{cred}}^{\mathsf{ask}_{\mathsf{sp}}}$.

Security of Sender Privacy. We give the security theorem of our sender privacy (SP) protocol. The proofs are given in the full version of the paper.

Theorem 3. *The SP protocol is sound if the q-SDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *in the random oracle model, where q is the maximum number of* Issue_e *oracle query. The SP protocol is unforgeable if the DL assumption holds in* $\mathbb{G}_1$ *in the random oracle model. The SP protocol is anonymous if the DDH assumption holds in* $\mathbb{G}_1$ *in the random oracle model.*

7 Performance Analysis

We analyze our PAChain in terms of throughput and latency, two of the most important metrics for analyzing the performance of a blockchain system. The latency of our PAChain is affected by the running time of the modules. The throughput of our PAChain is affected by both the running time of our three modules, and the size of each transaction.

7.1 Transaction Overhead

In this paper, we consider 128-bit security. The transaction amount is represented by a 64-bit positive integer (the same setting as Bitcoin and Monero).

For PAChain's transaction privacy, 64-bit of transaction amount implies that the range $R = 2^{64}$. We can take $\mathfrak{u} = 2^{16} = 65536$, $\ell = 4$. The public parameters for transaction privacy is about 2MB. The size of the ciphertext is 256 bytes. For each transaction output amount, the size of the range proof π_{enc} is 544 bytes [6].

[6] A 64-bit range proof by the recent Bulletproof [5] is about 800 bytes.

Table 2. Comparison of privacy-preserving blockchain schemes, for a standard 2-input-2-output transaction.

		Sender privacy	Recipient privacy	Tx privacy	Auditability	Authentication	Tx overhead (bytes)	Sender running time	Verifier running time
Public blockchain	Monero	◕	●	●			12704	300 ms	300 ms
	Zcash	●	●	●			576	120 s	10 ms
Consortium blockchain	Hyperledger Fabric	◔	◔	◔		●	628	10 ms	10 ms
	This paper	◕	●	●	●	●	2720	100 ms	100 ms

Table 3. Comparison for transaction privacy for a single output

	Setup time	C_{tp} Enc time	C_{tp} Dec time	π_{tp} Proof time	π_{tp} Verify time
Our scheme	53.8 s	2.8 ms	3.0 ms	27.1 ms	25.6 ms
Paillier encryption	402.6 ms	27.1 ms	7.4 ms		

The size of π_{tp} is 64 bytes plus all π_{enc} for all transaction outputs. For recipient privacy, the size of C_{rp} is 64 bytes, π_{rp} is 256 bytes for each recipient. The block randomness $R_{tx,r}$ is 32 bytes. (The 32 bytes of otpk_r replaces the output address and hence it is not viewed as an overhead). For sender privacy, the size of C_{sp} is 64 bytes, π_{rp} is 352 bytes and T is 32 bytes for each sender.

Considering a classical transaction of 2 inputs and 2 outputs, the overhead for privacy-enhancing consortium blockchain is 2720 bytes. We compare our PAChain with other schemes in Table 2:

- For consortium blockchain (e.g., Fabric or Corda), the classical transaction of 2 inputs and 2 outputs includes 2 ECDSA signatures from two inputs (128 bytes) and two X.509 certificates for 2 outputs' ECDSA public keys (about 500 bytes). The overhead is 628 bytes.
- For the public blockchain Monero, even if we consider the minimum ring size for ring signature as 3 (i.e., the real sender is one-out-of-three public keys. Hence the anonymity is very limited.), the total overhead is 12704 bytes for 2 inputs and 2 outputs.
- For Zcash, all the proofs can be combined to a single 288 bytes zk-SNARK proof. The total proof size becomes 576 bytes. However, the time for generating the proof will be much longer (> 120 s) and it requires a lot of RAM (> 3 GB). It causes a long latency in the blockchain system.

7.2 Module Implementation

We implemented our modules in a server with Intel Core i5 3.4GHz, 8GB RAM, running on Linux. Our implementation is by Golang, using BN256 pairing library.

Transaction Privacy. For transaction privacy, the running time for a single output is shown in Table 3. We compare our scheme with the additive homomorphic Paillier encryption with the same security level. When comparing with the encryption and decryption part only, our scheme is about 9 times and 2 times more efficient than the Paillier encryption. For the prover side, the complete transaction privacy is almost as efficient as a single Paillier encryption. Comparatively, our scheme takes a longer time for Setup, mainly for the generation of system parameters for the range proof.

Recipient Privacy. For recipient privacy for a single output, the Setup time is 4.6 ms, the CertIssue time is 1.4 ms, the Spend Time is 11.2 ms and the Verify time is 10.6 ms.

Sender Privacy. For sender privacy for a single input, the Setup time is 7.8 ms, the CredIssue time is 1.5 ms, the CredSign Time is 15.0 ms and the Verify time is 16.3 ms.

For a standard 2-input 2-output transaction, the total running time of our scheme (achieving all three properties) is 112 ms for the prover and 105 ms for the verifier side.

7.3 Testing Transaction Privacy with Hyperledger Fabric

We integrate the transaction privacy protocol in Hyperledger Fabric 1.0, in order to demonstrate our modulus can be consolidated into real world consortium blockchain. There are a few technical obstacles to implement our scheme. The first obstacle is that Fabric does not support optimization code of BN256 pairing written in C language. It results in > 10 times slower exponentiation and pairing computation. We expect future version of Fabric to allow optimization for pairing-based computation.

The second difficulty is to implement the verification logic into the smart contract (chaincode) of Fabric. We built a complete flow of transaction, including the creation of money (deposit), normal transaction, balance query and the destroy of money (withdraw). The chaincode has 2223 lines of codes. The extra codes for server side and client are 575 lines and 1061 lines respectively. The common module has 823 lines. (Comparatively, the core transaction privacy protocol has 2143 lines of codes.)

Transaction Privacy. In our current implementation for a 2-input 2-output transaction in Hyperledger Fabric 1.0, the signing time is 988 ms and the verification time is 1.35 s. Our implementation shows that other processing time for the transaction packet in negligible when compared to cryptographic operations. We expect that if optimization code of pairing is allowed, the signing and verification time can be about 100 ms.

The consensus algorithm is the current bottleneck of most consortium blockchain systems. The PBFT consensus algorithm used in Hyperledger Fabric 1.0 allows about 2000 transactions per second and has about 1 s of latency. If optimization is allowed in Fabric, our scheme has a running time of 100 ms for both the prover and verifier side, for a standard 2-input 2-output transaction. Therefore, our scheme is practical and will not become the bottleneck of the consortium blockchain system.

8 Conclusion

In this paper, we propose efficient solution for privacy, auditability and authentication in consortium blockchain. We give module solutions for them, so that they can be added to blockchain according to actual business need. We implemented our schemes and they are more efficient than the existing solutions in public blockchain.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006). https://doi.org/10.1007/11832072_8
2. Ben-Sasson, E., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: IEEE SP 2014, pp. 459–474. IEEE Computer Society (2014)
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_4
4. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_3
5. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: IEEE SP 2018, pp. 315–334. IEEE (2018). https://doi.org/10.1109/SP.2018.00020
6. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_15
7. Camenisch, J., Mödersheim, S., Sommer, D.: A formal model of identity mixer. In: Kowalewski, S., Roveri, M. (eds.) FMICS 2010. LNCS, vol. 6371, pp. 198–214. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15898-8_13
8. Garman, C., Green, M., Miers, I.: Accountable privacy for decentralized anonymous payments. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 81–98. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54970-4_5
9. Li, W., Sforzin, A., Fedorov, S., Karame, G.O.: Towards scalable and private industrial blockchains. In: BCC 2017, pp. 9–14. ACM (2017)
10. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009). https://bitcoin.org/bitcoin.pdf
11. Noether, S.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098 (2015). http://eprint.iacr.org/
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16

13. Ruffing, T., Moreno-Sanchez, P., Kate, A.: CoinShuffle: practical decentralized coin mixing for bitcoin. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 345–364. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_20
14. 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
15. Wüst, K., Kostiainen, K., Capkun, V., Capkun, S.: Prcash: Centrally-issued digital currency with privacy and regulation. In: FC 2019, Cryptology ePrint Archive, Report 2018/412 (2018). https://eprint.iacr.org/2018/412