PPChain: A Privacy-Preserving Permissioned Blockchain Architecture for Cryptocurrency and Other Regulated Applications

Chao Lin, Debiao He, Xinyi Huang, Xiang Xie, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—Existing (popular) blockchain architectures, including the widely used Ethereum and Hyperledger, are generally not designed to achieve conflicting properties such as anonymity and regulation, and transparency and confidentiality. In this article, we propose a privacy-preserving permissioned blockchain architecture (PPChain) that permits one to also introduce regulation, where PPChain's architecture is modified from that of Ethereum. Specifically, we integrate the cryptographic primitives (group signature and broadcast encryption), and adopt practical byzantine fault tolerance consensus protocol with a validate-record separation mechanism, as well as removing the transaction fee and mining reward. To show the utility of PPChain, we provide qualitative security and privacy analysis, and performance analysis. We also explain how PPChain can be deployed in regulation applications, using cryptocurrency, food supply chain, and sealed-bid auctions as examples.**

***Index Terms*—Permissioned blockchain, practical byzantine fault tolerance (PBFT), privacy preserving, privacy-preserving permissioned blockchain architecture (PPChain), regulation.**

Manuscript received April 2, 2020; revised July 18, 2020; accepted August 25, 2020. Date of publication September 17, 2020; date of current version August 26, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802500, in part by the National Natural Science Foundation of China under Grants 61972294, 61932016, 61822202, 61772377, and 61841701, in part by the Natural Science Foundation of Hubei Province of China under Grant 2020CFA052, and in part by the Wuhan Municipal Science and Technology Project under Grant 2020010601012187. The work of Kim-Kwang Raymond Choo was supported by the Cloud Technology Endowed Professorship. *(Corresponding author: Debiao He.)*

Chao Lin is with the Fujian Provincial Key Laboratory of Network Security and Cryptology/Center for Applied Mathematics of Fujian Province, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350007, China, and also with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: linchao91@qq.com).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: hedebiao@163.com).

Xinyi Huang is with the Fujian Provincial Key Laboratory of Network Security and Cryptology/Center for Applied Mathematics of Fujian Province, College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China (e-mail: xyhuang81@gmail.com).

Xiang Xie is with the Juzix Technology Co. Ltd., Shenzhen 518000, China (e-mail: xiexiang@juzix.net).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JSYST.2020.3019923

I. Introduction

BLOCKCHAIN is a distributed ledger that records transactions, which are chained by a hash in chronological order [1]–[3]. It is commonly maintained by some mutually untrusting nodes, and every node possesses a copy of the ledger. A consensus protocol is required for these nodes to achieve consistency, that is, verifying transactions, packaging transactions into blocks, and constructing a hash chain over blocks. There are different consensus protocols, such as proof of work (PoW), proof of stake (PoS), and practical byzantine fault tolerance (PBFT), for different types of blockchain (i.e., public, permissioned, and private) [4].

In a public blockchain, anyone can freely participate in or terminate the generation of blocks based on classical protocols (e.g., PoW and PoS). Such blockchain generally involves a native cryptocurrency (e.g., Bitcoin and Ethereum [5]). Permissioned and private blockchains, on the other hand, are maintained by a set of known, reliable participants [also referred to as permissioned nodes (PNs)] based on traditional protocols (e.g., PBFT and a consensus algorithm for managing a replicated log, namely, RAFT). In permissioned blockchain (e.g., Hyperledger [6]), only users who have been authorized by PNs can join the blockchain, unlike a private blockchain.

As discussed, cryptocurrency is natively supported by public blockchain, which is the most mature public blockchain application to date. Moreover, for the execution of arbitrary, programmable transaction logic as smart contract (e.g., in Ethereum) or chaincode (e.g., in Hyperledger), the blockchain can function as a trusted distributed entity. Thus, emerging applications, such as Internet of Things, blockchain government, and identity management, can leverage blockchain (mostly permissioned or private blockchain) [7]–[10]. However, striking a balance between conflicting properties such as anonymity and regulation, and transparency and confidentiality, in existing blockchain systems (e.g., Ethereum and Hyperledger) remains challenging.

1) *Anonymity and Regulation*: Most popular blockchain systems, such as Bitcoin, Ethereum, and Hyperledger, use the elliptic curve digital signature algorithm (ECDSA) for issuing transactions. These systems can only provide simple unlinkability, due to the use of pseudonyms a user's real identity will be revealed in the event of successful attacks such as network analysis, address clustering and transaction graph analysis [11]–[13]. To enhance the anonymity of blockchain, a number of different protocols have been proposed (e.g., Zerocash [11], Monero [12], Mimblewimble [13], and QuisQuis [14]). However, it is also known

that completely anonymous systems can be exploited for criminal or malicious activities such as receiving proceeds of crime (e.g., drug/weapon trafficking, ransomware, and sex trafficking and child sexual exploitation). Thus, how to strike a balance between achieving anonymity and minimizing the need for regulation remains an operational and research challenge.

While there have been attempts to determine an optimal balance between the two, introducing regulation can incur (significant) costs, for example, in terms of communication or computation overheads. For instance, a complex signature of knowledge (SoK) is required in Traceable Monero [15] when generating a traceable one-time public key. In addition, there are considerable interactions (about 14 times) and a number of cryptographic operations involved [16]. These complex and expensive communication and computing costs complicate real-world deployment, particularly as the system becomes more popular, i.e., scalability becomes an issue.

2) *Transparency and Confidentiality*: In early blockchain systems (e.g., Bitcoin and Ethereum), all internal states of the ledger are publicly visible for verifiability and final consistency. While such transparency (in message deliveries and computations) is appreciated for a number of reasons, there are also data privacy concerns. As a countermeasure, we can utilize existing cryptographic techniques, ranging from data encryption to advanced zero-knowledge proofs to verifiable computation, and so on. However, there are associated costs (e.g., computational and communication overheads), and hence this is not practically viable [6].

   To solve this issue, Hyperledger [6] adopts an execute-order-validate paradigm, and achieves not only confidentiality but also resiliency, flexibility and scalability. This paradigm also allows for parallel execution of transactions. However, due to its unsupported sequential execution, it is no long suitable for cryptocurrency projects.[1]

Contributions: This article proposes a privacy-preserving permissioned blockchain architecture (PPChain) to mitigate the aforementioned tensions. Specifically, on basis of Ethereum architecture, we modify the transaction design by integrating group signature (GS) and broadcast encryption (BE), and further adopt PBFT with a validate-record separation mechanism for the consensus. PPChain can be not only used in developing regulated cryptocurrency, but also suitable for many privacy-preserving applications such as, but not limited to, food supply chain management and sealed-bid auctions. The key new features of PPChain are described as follows.

1) An anonymous transaction design with functionality of regulation, but avoiding the existing complex interaction and cryptographic operations (i.e., SoK).
2) A point-to-multi-point data transmission method that realizes both data confidentiality and verifiability.
3) A native cryptocurrency (PPCoin) inherits the account model[2] of Ethereum, but removes the transaction fee and mining reward due to using PBFT consensus protocol. It is worth to note that we maintain the gas[3] to avoid network abuse or inevitable problems caused by Turing completeness, which is indeed different from the existing permissioned blockchains such as Hyperledger.
4) A validate-record separation mechanism that mitigates the limited throughput caused by sequential execution of transactions in existing order-execute architecture such as Ethereum.

The rest of this article is arranged as follows. On basis of system definitions including cryptography materials, network model, system components, and security model presented in Section III, we construct our PPChain and its cryptocurrency application in Section IV. In Section V and Section VI, we provide the evaluation and discussion, potential applications, respectively, to demonstrate the utility of our PPChain. Finally, Section VII concludes this article.

II. Related Work

To enhance the privacy protection of blockchain systems, some mixing technology-based solutions have been proposed (e.g., Onionbc,[4] Bitmixer[5]). This technology unlinks the relationship among participants through shuffling transaction inputs and outputs, but it is faced with some known limitations (e.g., theft of digital asset, disclosure of participant privacy by a malicious central mixing node) [17]. To address the former problem, Maxwell [18] introduced a conditional mixing mechanism (i.e. CoinSwap). CoinSwap uses a hashing lock to build the connection between escrow transactions and corresponding redeeming ones, and hence it can prevent the central mixing node from stealing the participant's assets. Bonneau *et al.* [19] also proposed a modified Mixcoin by using a signature-based accountability technology, which can effectively track misbehavior of the central mixing node. Duffiel and Diaz [20] tried to address the privacy leakage issue via an economical method, that is, the central mixing node needs to prepay a deposit before merging multiple transactions. This deposit will be lost once the central mixing node does some unauthorized or illegal operations.

For solving the latter problem (i.e., the mixing node may compromise the participants' privacy), Valenta and Rowan [21] integrated blind signatures into Mixcoin and designed a novel solution (named as Blindcoin). In the Blindcoin, the central mixing node only provides the mixing services but not be able to link the transaction inputs and outputs. Unfortunately, the Blindcoin requires that the mixed amount is fixed and participants need to anonymously share the transaction output, which cannot avoid third-party cheating. In addition, these central mixing solutions also suffer from long waiting delay, additional mixing costs, single point failure of central mixing node, and other limitations. Although some decentralized mixing solutions such as CoinShuffle [22], XIM [23], and CoinParty [24] have been proposed to avoid single point of failure, and achieve higher anonymity, there still exist limitations such as longer waiting delays and higher computation costs.

[1]While one may be able to develop a native digital token, it will be a complex development project requiring significant time and effort.

[2]There are two types of record-keeping models in existing blockchain, that is, Unspent Transaction Output (UTXO) Model and Account Model. In the first mode, all of the unspent transactions are kept in each fully synchronized node. But the second one keeps track of the balance of each account as a global state, an analogy of which is using an ATM/debit card.

[3]The concept of gas was introduced in Ethereum to indicate the consumption toward computational expenses. It refers to the fee or pricing value, required to successfully issue a transaction or invoke a contract on the Ethereum.

[4]OnionBC is an anonymous online wallet, available at: https://bitcointalk.org/index.php?topic=288099.0

[5]Bitmixer is one of the largest Bitcoin mixing services, available at: https://bitcointalk.org/index.php?topic=415396.160

Hence, there are various cryptography-based schemes have been proposed. Monero[6] adopts ring signatures and one-time payment mechanism to achieve the anonymity of sender and unlinkability between the sender and receiver. It can ensure that only some unknown number of coins moved from one of these input addresses to a one-time address can be obtained. However, Kumar *et al.* [25] stressed that attackers can still link transaction address to the real identity when the size of anonymous set in the ring signature is small. To guarantee its unlinkability, a large anonymous set can be chosen, but it will significantly increase the storage costs for transactions. To deal with this contradiction, Sun *et al.* [12] adopted accumulator tools to design a novel ring signature mechanism (i.e., RingCT 2.0), which can balance the efficiency and privacy concerns in Monero.

To mitigate the requirement of redundant addresses in Monero, Miers *et al.* [26] integrated commitment and zero-knowledge proof to propose a novel solution (i.e., Zerocoin). The commitment can encapsulate transaction inputs and outputs, and the zero-knowledge proof can further prove the transactions without revealing the correlation. Nevertheless, Zerocoin only supports transactions of fixed denominations, and users who have taken over the transaction can track its flow. Then, Ben-Sasson *et al.* [11] used zero-knowledge succinct noninteractive argument of knowledge to propose Zerocash. In Zerocash, users first place the coins into a "shielded pool" and then they can prove the right of spending these coins in a zero-knowledge way. Only the user's right can be verified, the knowledge of spending which particular coins has been hided. Although Zerocash can hide users' identities and support transactions of any denomination, there exist some limitations among it such as generating system parameters under trusted nodes, slow proving process, and lack of any regulation / auditing capability. Solutions such as [13], [27], and [28] were also proposed to achieve privacy protection in blockchain systems, but few of them consider regulation which is indispensable for minimizing abuse/criminal exploitation.

To balance privacy protection and regulation in blockchain systems, Li *et al.* [15] integrated ElGamal encryption, commitment, and zero-knowledge proof into the Monero system for a traceable Monero system. Wu *et al.* [16] used blind signatures and key derivation mechanism to propose a concrete system. Lin *et al.* [29] used self-updating pseudonym technology and SoK to construct a more efficient conditional anonymous payment system. However, these solutions only support the UTXO model but fail to be applied in the account model which is with more wide application prospect. In addition, they either require significant communication and computation overheads (e.g., a complex SoK for tracing in [15], considerable interactions and cryptographic operations in [16]), or have ambiguous mission from the view of managers (i.e., managers also need to maintain the blockchain ledger in [29]).

III. System Definitions

A. Cryptography Materials

1) Elliptic Curve Digital Signature Algorithm: The existing blockchain systems (e.g., Bitcoin and Ethereum) mainly adopt the ECDSA [30], [31] to sign the raw transaction content. The ECDSA has been proven secure by Brown [30] under the assumption of collision resistance of hash function. It consists of the following algorithms.

1) ESetup(λ): The setup algorithm takes as input a security parameter λ, and it returns system public parameters $PP = (E, \mathbb{G}, P, p, q, a, b, \mathcal{H})$, where p, q is primer numbers of λ bits, E is a nonsingular elliptic curve defined by $y^2 = x^3 + ax + b(modp)(a, b \in \mathbb{F}_p)$, $\mathbb{G}$ is an additive group consisting of all the points of E and an infinity point O, P is a generator of $\mathbb{G}$ of order q, and $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a secure hash function.
2) EKG(PP): This key generation algorithm takes as input system public parameters PP, and it randomly chooses $x \in \mathbb{Z}_q$ to compute $Y = xP$. It finally returns a public key $pk_s = Y$ and corresponding secret key $sk_s = x$.
3) ESign(PP, sk_s, m): This signing algorithm takes as input system public parameters PP, a secret key $sk_s = x$ and a message m. It randomly chooses $k \in \mathbb{Z}_q$ to compute $(r_x, r_y) = kP(modq)$, $s = k^{-1} \cdot (\mathcal{H}(m) + x \cdot r_x)(modq)$, and returns a signature $\sigma = (r_x, s)$ of the message m.
4) EVerify(PP, pk_s, m, σ): This verification algorithm takes as input system public parameters PP, a public key $pk_s = Y$, a message m, and a candidate signature σ. It parses σ as (r_x, s), and computes $(r_x,' r'_y) = (s^{-1} \cdot \mathcal{H}(m))P + (s^{-1} \cdot r_x)Y$. If $r'_x = r_x(modq)$, it returns 1 to indicate the validity of σ; otherwise, it returns 0.

2) Broadcast Encryption: BE is a type of encryption primitives that can build a secure communication between one sender and multiple preselected receivers [32], [33]. By using the BE, a single sender can securely broadcast the same message to multiple receivers over an open network. The BE owns the confidentiality of messages and anonymity of receivers, thus we can adopt the BE to protect the privacy of transaction data in blockchain system. Although we use the following BE scheme proposed in [33] for the instantiation, any secure and efficient BE scheme can be adopted in our proposal.

1) BSetup(λ): This setup algorithm is invoked by a key generation center (KGC) to generate some system public parameters and a master secret key. It takes as input a security parameter λ, and returns system public parameters $PP = \{E, \mathbb{G}, p, q, a, b, P, h, \mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, P_0\}$, where p, q is primer numbers of λ bits, E is a nonsingular elliptic curve defined by $y^2 = x^3 + ax + b(modp)(a, b \in \mathbb{F}_p)$, $\mathbb{G}$ is an additive group consisting of all the points of E and an infinity point O, P is a generator of $\mathbb{G}$ of order q, $h = \#E(\mathbb{F}_p)/q$ is a complementary factor, $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ are four secure hash functions, $P_0 = sP$ is the system master public key (where $s \in \mathbb{Z}_q^*$). The corresponding secret is $msk = s$, which is secretly stored by the KGC.
2) BSVG(PP): This private information generation algorithm is invoked by a user to generate his / her partial secret and public keys. Suppose that the identification of user is ID_i, then this algorithm takes as input system public parameters PP, and randomly chooses $x_i \in \mathbb{Z}_q^*$ to compute $P_i = x_i P$. It finally returns the user's partial secret key x_i and partial public key P_i.
3) BPPKG(PP, ID_i, P_i): This partial private key generation algorithm is invoked by the KGC to generate the other partial keys. It takes as input system public parameters PP and a user's public information (ID_i, P_i), it randomly chooses $t_i \in \mathbb{Z}_q^*$ to compute $T_i = t_i P$, $l_i = \mathcal{H}_0(ID_i, T_i, P_i)$, $d_i = (t_i + sl_i)(modq)$, anf returns the

[6]Monero is one type of anonymous cryptocurrencies, available at: https://www.getmonero.org/

user's other partial secret key d_i and other partial public key T_i. The KGC sends the (d_i, T_i) to the user through a secure channel.

4) BKG(PP, ID_i, P_i): This key generation algorithm invokes the algorithms of BSVG and BPPKG to obtain a user's secret key $sk_i = (d_i, x_i)$ and public key $pk_i = (P_i, T_i)$.
5) BEnc$(PP, (pk_1, \ldots, pk_t), m)$: This encryption algorithm is invoked by a sender S to generate a broadcast ciphertext of a plaintext m, where the broadcast ciphertext can be decrypted by t-receivers $R_1, R_2, \ldots, R_t$. It takes as input system public parameters PP, the t-receivers' public keys $pk_1, \ldots, pk_t$ and the plaintext m. First, it randomly chooses $r \in \mathbb{Z}_q^*$ to compute $R = rP$, $l_i = \mathcal{H}_0(ID_i, T_i, P_i)$, $u_i = r(T_i + l_i P_0 + P_i)$, $\alpha_i = \mathcal{H}_1(R, u_i)$, where $i = 1, 2, \ldots, t$. Then, it randomly chooses $\theta \in \mathbb{Z}_q^*$ to compute $f(x) = \Pi_{i=1}^{t}(x - \alpha_i) + \theta(modp) = x^t + c_{t-1}x^{t-1} + \cdots + c_1 x + c_0 (c_i \in \mathbb{Z}_q^*)$, $\beta = \mathcal{H}_2(R, \theta)$, $\delta = AES.Enc_\beta(m)$, $\tau = \mathcal{H}_3(c_0, c_1, \ldots, c_{t-1}, m, \theta, R, \delta)$. Finally, it returns the ciphertext $c = (c_0, c_1, \ldots, c_{t-1}, R, \delta, \tau)$.
6) BDec(PP, sk_i, c): This decryption algorithm is invoked by the t-receivers $\mathcal{R}_i$ to decrypt a ciphertext. It takse as input system public parameters PP, a secret key sk_i and a ciphertext $c = (c_0, \ldots, c_{t-1}, R, \delta, \tau)$. Then, it computes $u_i = (d_i + x_i)R$, $\alpha_i = \mathcal{H}_1(R, u_i)$, $f(x) = x^t + c_{t-1}x^{t-1} + \cdots + c_1 x + c_0$, $\theta = f(\alpha_i)$, $\beta = \mathcal{H}_2(R, \theta)$, $m = AES.Dec_\beta(\delta)$, and $\tau' = \mathcal{H}_3(c_0, \ldots, c_{t-1}, m, \theta, R, \delta)$. Finally, it returns the plaintext m if $\tau' = \tau$, or returns $\perp$ representing the c is invalid.

3) Group Signature: GS allows any group member to sign a message without revealing their identities [34]. The group public key and group members' private keys are uniformly generated by a group manager, which ensures that only the group manager can identify the real identity of a malicious group member during the dispute. In this article, we apply the GS for achieving the anonymity and regulation of blockchain systems. Specifically, we adopt the GS scheme proposed in [35] for the instantiation, but any other secure and efficient GS schemes can also be used in our proposal.

1) GSetup(λ): This setup algorithm takes as input a secure parameter λ, and first generates system public parameters $PP = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, \mathcal{H}(\cdot), \mathcal{H}_4(\cdot))$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three cyclic groups of prime order q (where q is λ bits), $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pair, P_1 is a generator of $\mathbb{G}_1$, P_2 is a generator of $\mathbb{G}_2$, $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ and $\mathcal{H}_4 : \{0,1\}^* \to \{0,1\}^{256}$ are secure hash functions. Then, it randomly chooses $d, s, u \in \mathbb{Z}_q^*$ to compute $D = d \cdot P_1, S = s \cdot P_2, U = u \cdot P_1$. Finally, it returns system public parameters PP, a master secret key of group manager $sk = (d, s)$, a tracing key $tk = u$ and group public key $gpk = (D, S, U)$.
2) GEnroll(PP, sk): This enrolling algorithm takes as input system public parameters PP and master secret key of group manager $sk = (d, s)$. It randomly chooses $x_i \in \mathbb{Z}_q^*$ to compute $Z_i = (d - x_i)(sx_i)^{-1} \cdot P_1$ and $tag_i = \mathcal{H}_4(x_i \cdot Z_i)$. Finally, it returns the group member's secret key $gsk_i = (x_i, Z_i)$ and a tag tag_i. To trace the real identity of group member, the group manager needs to maintain a member list $List = (ID_i, GU_i, tag_i)$, where ID_i is the real identity of group member GU_i.
3) GSign(PP, gsk_i, gpk, msg): This GS generation algorithm takes input system public parameters PP, a group member's secret key gsk_i, group public key $gpk = (D, S, U)$ and a message msg. It randomly chooses $k \in \mathbb{Z}_q^*$ to compute $C_1 = k \cdot P_1$, $C_2 = x_i \cdot Z_i + k \cdot U$, $Q = e(U, S)^k$, $digest = \mathcal{H}_4(msg)$, $c = \mathcal{H}(C_1, C_2, Q, digest)$ and $w = kc + x_i(modq)$, and returns the GS (C_1, C_2, c, w) of message msg.
4) GVerify(PP, msg, σ, gpk): This GS verification algorithm takes as input system public parameters PP, a candidate message-signature pair (msg, σ) and group public key $gpk = (D, S, U)$. It computes
$$\widehat{Q} = \frac{e(C_2, S) \cdot e(w \cdot P_1, P_2)}{e(c \cdot C_1 + D, P_2)}, digest = \mathcal{H}_4(msg)$$
and verifies the validity of (msg, σ) through checks if the equation $c \stackrel{?}{=} \mathcal{H}(C_1, C_2, \widehat{Q}, digest)$ holds on.
5) GTrace(PP, σ, tk): This tracing algorithm takes as input system public parameters PP, GS $\sigma = (C_1, C_2, c, w)$, and a tracing key $tk = u$. It computes
$$tag_i = \mathcal{H}_4(x_i \cdot Z_i) = \mathcal{H}_4(C_2 - u \cdot C_1)$$
and searches the real identity of the signer from the member list $List = (ID_i, GU_i, tag_i)$.
6) GRev$(info)$: This permission revocation algorithm is invoked by the group manager to revoke the authority of a group member. It takes as input a group member's private information $info = x_i \cdot Z_i$, and then it stores $info$ into the revocation list $revoList$ to revoke the authority.

B. Network Model

The participants in our PPChain include registration center (RC), user, manager, and PN (see Fig. 2), which constitute the permissioned blockchain ecosystem.

1) *RC*: This participant is a trusted entity, who is in charge of other participants' registers. Specifically, any user, manager, or PN needs to obtain register information (i.e., key information of GS and BE) from RC before it joins our PPChain.
2) *User*: These participants are the main stakeholder group, who consume the services provided by PPChain such as regulated cryptocurrency, food supply chain, and sealed-bid auctions. Each user has an account address and private key,[7] the certificate of account address, group private key, and decryption private key, which are necessary in issuing transactions for achieving security, anonymity and traceability. The concrete usage of these keys will be detailed later.
3) *Manager*: This participant is also a trusted entity, but which only takes charge of tracing malicious behavior and authority revocation. Hence, in addition to owning the above keys as a user, the manager has an additional tracing key for performing its responsibility.
4) *PN*: These participants are some authorized entities, who are responsible for processing and maintaining the PPChain ledger based on a consensus protocol (i.e.,

[7]In blockchain systems (e.g., Bitcoin and Ethereum), an address is generally derived from a public key using some secure hash function (e.g., RIPEMD-160). We also adopt this mechanism for our architecture, but for the purposes of this article, we simply describe the public key as the address.

PBFT). To deal with the limited throughput caused by sequential execution of transactions in Ethereum, we adopt a validate-record separation mechanism. That is, PNs in PPChain are classified into "Validating Node" and "Recording Node," where the former is responsible for verifying transactions and the latter recording validated transactions into the PPChain. We remark that this idea is inspired by Hyperledger, which can efficiently improve the efficiency of consensus.

C. System Components

A PPChain system consists of the following phases.

1) $PP \leftarrow Initialize(1^{\lambda})$: This initialization phase takes as input a security parameter λ, and returns a public system parameter PP.
2) $(apk, ask, \mathbf{pk}, \mathbf{sk}) \leftarrow Register(PP)$: This phase is used for the participants to register their account address apk and private key ask, as well as other related public keys $\mathbf{pk}$ and private keys $\mathbf{sk}$.
3) $tx \leftarrow Transact(num, ask_s, \mathbf{sk}_s, apk_s, apk_r)$: The user executes this phase to issue a transaction, that is, it takes as input a transferred value num, its private keys ask_s and $\mathbf{sk}_s$, its account address apk_s and its target address apk_r, and returns a transaction tx.
4) $\{0, 1\} \leftarrow Chain(tx)$: In this chaining phase, on basis of the PBFT consensus mechanism, the PNs chain the valid transaction tx requested from the user.
5) $rid \leftarrow Trace(\mathbf{sk}_m, tx)$: The manager can trace the real identity of any malicious transaction in this phase. That is, it takes as input its private key $\mathbf{sk}_m$ and the malicious transaction tx, returns the real identity of the user who has issued the tx.

D. Security Model

We now formalize the security model, including anonymity, confidentiality and traceability, definitions of which are described as follows.

Definition 1: Anonymity. This property requires that the ledger and transaction reveal no real identity of issuer to the adversary, except publicly available information (e.g., some public keys), that is, a PPChain is anonymous if for all $\mathcal{PPT}$ distinguishers $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2\}$, it satisfies (1) shown at the bottom of this page, where we require that both the transactions tx_0 and tx_1 have not been traced, and define the oracles $\mathcal{O}_r$, $\mathcal{O}_{tt}$, $\mathcal{O}_c$, and $\mathcal{O}_{te}$ as follows.

1) $\mathcal{O}_r(i)$: Given a query number i, it generates an address and private key (apk_i, ask_i), and initializes some coins in apk_i. Then, it computes the additional public keys $\mathbf{pk}_i$ and private keys $\mathbf{sk}_i$. Finally, it adds i and $(apk_i, ask_i, \mathbf{pk}_i, \mathbf{sk}_i)$ into the list $\mathcal{L}$ locally and returns $(apk_i, \mathbf{pk}_i)$. Assuming that the lth account is the manager (i.e., $(apk_l, ask_l, \mathbf{pk}_l, \mathbf{sk}_l, gtk)$) for the response of $Tracing$, where gtk is the tracing key.
2) $\mathcal{O}_{tt}(num, apk_s, apk_r)$: Given a transaction value num, sender's account apk_s and receiver's account apk_r, it invokes $tx \leftarrow Transact(num, ask_s, \mathbf{sk}_s, apk_s, apk_r)$. It then broadcasts tx to blockchain network for being chained by PNs.
3) $\mathcal{O}_c(i)$: Given a query number $i \in \mathcal{L}$, it retrieves the list $\mathcal{L}$ to return $(apk_i, ask_i, \mathbf{pk}_i, \mathbf{sk}_i)$.
4) $\mathcal{O}_{te}(tx)$: Given a transaction, it uses the manager's tracing key gtk to return the signer's real identity of the tx.

Definition 2: Confidentiality: This property guarantees that the ledger and transaction will not disclose any transaction value, that is, a PPChain owns the confidentiality if for all $\mathcal{PPT}$ adversaries $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$, it satisfies constraint (2) shown at the bottom of this page, where all the oracles (except the $\mathcal{O}_{te}$) are identical to the above definitions and the modified $\mathcal{O}_{te}$ is changed to return the original transaction information with the help of PNs. In addition, we require that both the account apk_s and apk_r have not been corrupted.

Definition 3: Traceability. This property is provided to trace the sender's real identity of a malicious transaction. We say a PPChain owns this property if there is a traceability algorithm `Trace` shown in (3) at the the bottom of this page.

$$\left| \Pr \begin{pmatrix} PP \leftarrow Initialize(1^{\lambda}); (apk_m, ask_m, \mathbf{pk}_m, \mathbf{sk}_m, gtk) \\ \leftarrow Register(PP); \{tx_i\} \leftarrow \mathcal{D}_1^{\mathcal{O}_r, \mathcal{O}_{tt}, \mathcal{O}_c, \mathcal{O}_{te}}(PP), \\ i = \{0, 1\}; b \leftarrow \{0, 1\}; rid_b \leftarrow Trace(\mathbf{sk}_m, tx_b); \\ b' \leftarrow \mathcal{D}_2^{\mathcal{O}_r, \mathcal{O}_{tt}, \mathcal{O}_c, \mathcal{O}_{te}}(PP, rid_b) : b' = b \end{pmatrix} - \frac{1}{2} \right| \leq \text{negl}(\lambda) \tag{1}$$

$$\left| \Pr \begin{pmatrix} PP \leftarrow Initialize(1^{\lambda}); (num_0, num_1, apk_s, apk_r) \\ \leftarrow \mathcal{A}_1^{\mathcal{O}_r, \mathcal{O}_{tt}, \mathcal{O}_c, \mathcal{O}_{te}}(PP); b \leftarrow \{0, 1\}; \\ tx_b \leftarrow Transact(num_b, ask_s, \mathbf{sk}_s, apk_s, apk_r); \\ b' \leftarrow \mathcal{A}_2^{\mathcal{O}_r, \mathcal{O}_{tt}, \mathcal{O}_c, \mathcal{O}_{te}}(PP, tx_b) : b' = b \end{pmatrix} - \frac{1}{2} \right| \leq \text{negl}(\lambda) \tag{2}$$

$$\left| \Pr \begin{pmatrix} PP \leftarrow Initialize(1^{\lambda}); (apk_m, ask_m, \mathbf{pk}_m, \mathbf{sk}_m, gtk) \\ \leftarrow Register(PP); \forall i \in \mathcal{L}, (apk_i, ask_i, \mathbf{pk}_i, \mathbf{sk}_i) \leftarrow \\ Register(PP); \forall a_l, b_l \in \mathcal{L}, tx_l \leftarrow Transact(num_l, \\ ask_{a_l}, \mathbf{sk}_{a_l}, apk_{a_l}, apk_{b_l}) : Trace(PP, tx_l) = rid_l \end{pmatrix} \right| = 1 \tag{3}$$

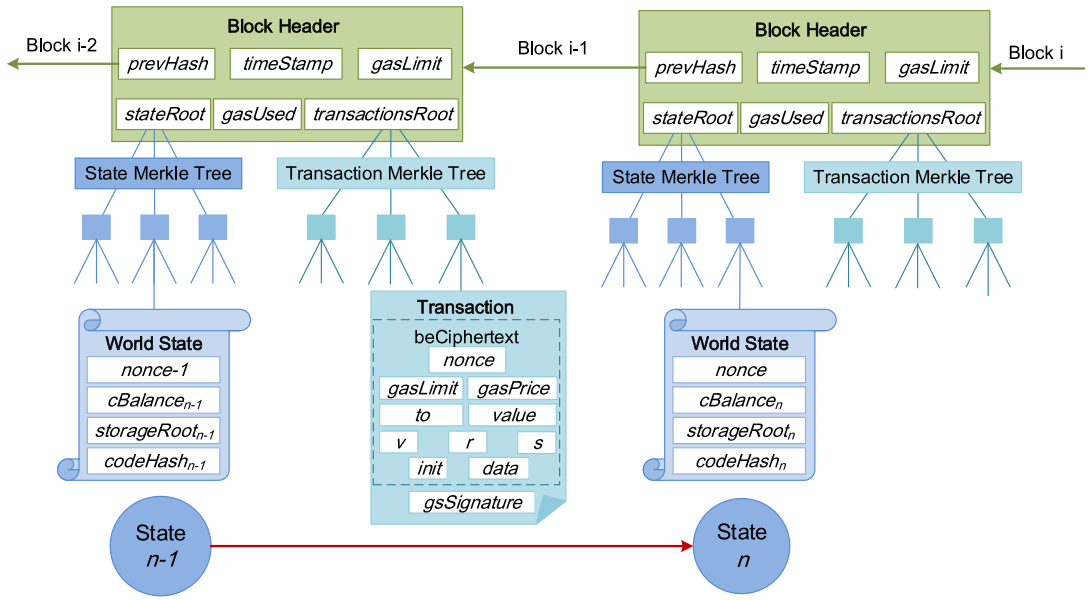


Fig. 1. Architecture of PPChain.

IV. PPCHAIN: PRIVACY-PRESERVING PERMISSIONED BLOCKCHAIN SYSTEM

A. Overview of PPChain Architecture

Our PPChain, like Ethereum [5], can also be regarded as a transaction-based state machine that begins with a genesis state and incrementally executes transactions to some final (current) state. In the architecture of PPChain (see Fig. 1), transactions are between two states and collected into the block that stores tree roots associated with the state and the transaction in the block header. The current block is linked to the previous one using a cryptographic hash. Compared to Ethereum, our PPChain makes the following changes to become a privacy-preserving permissioned blockchain architecture with regulation.

1) *Consensus*: The incentive protocol in public blockchain (e.g., Ethereum) can attract more miners to join and hence enhances the security of blockchain network. However, this is meaningless for permissioned blockchain, where the ledger is maintained by some small-scale PNs. Hence, we replace PoW with PBFT for the consensus of PPChain ledger. This allows us to not only improve the consensus efficiency, but also avoid wastage of power resources.
2) *Transaction*: Each transaction in the PPChain needs to be cryptographically signed (i.e., using ECDSA signature) by the account, but it is additionally embedded with BE and GS. It means that a user who would like to issue a transaction, after preparing a transaction like Ethereum, must use BE to encrypt and then sign this transaction using GS. It is worth to note that the ECDSA signature cannot be removed for which is necessary for the original Ethereum transaction. The specific group member's address cannot be conformed if we only use the GS, because all the group members share the same group public key.
   In the PPChain, we remove the transaction fee, but maintain the gas to control the use of resources (i.e., avoiding network abuse and inevitable problems caused by Turing completeness). For a general transaction, the user does not pay for the gas; but for a message call transaction (involving an invocation of smart contract), the user has to pay for the gas. Correspondingly, each transaction and block will set an upper limit of the max gas (i.e., gas limit).
3) *World State*: We also adopt a world state to map account address to its state, which is stored in a state Merkle tree. In the PPChain, the world state also consists of four fields ($nonce$, $cBalance$, $storageRoot$, $codeHash$), but the $cBalance$ here differs from that of Ethereum (i.e., the balance in PPChain is encrypted under the BE). That is, only the account itself and PNs can decrypt the $cBalance$ to obtain the original balance.

Note that our PPChain inherits the functionality of smart contract supported by Ethereum. It is well known that operations of smart contract (including deployment and invocation) are all executed through issuing transactions. Our PPChain can preserve the privacy of each transaction, and hence the privacy of smart contract can also be provided. The only price is the computation or communication costs caused by performing GS and BE schemes.

Thus, the contract in our PPChain can also be programmed in a Turing complete language (e.g., Solidity), by using which we can deploy abundant privacy-preserving applications (e.g., regulated cryptocurrency, digital right management, and payment channels) in PPChain. There are also two types of transaction in PPChain, namely: "message call" and "contract creation." In the contract creation transaction, the field of to is set as null and the $init$ is set as an EVM-code fragment. In contrast, a message call transaction specifies the $data$ to invoke the algorithms in smart contract.

B. PPChain for Regulated Cryptocurrency

Building on the aforementioned architecture of PPChain, we will now construct a regulated cryptocurrency (hereafter referred to as PPCoin), and then discuss how this PPCoin can achieve the properties of anonymity, confidentiality, and traceability. Note

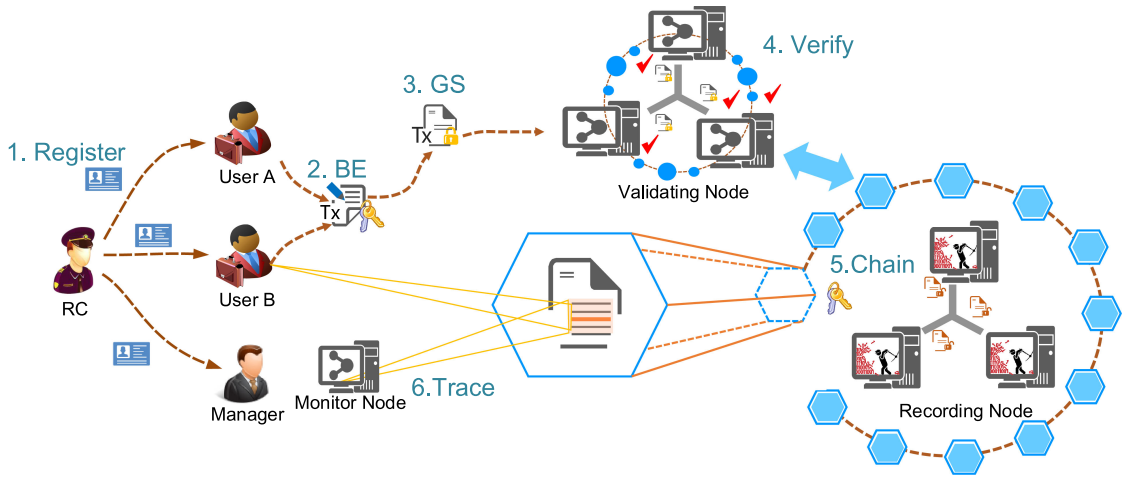


Fig. 2. Deign of PPCoin (a regulated cryptocurrency based on PPChain).

that, in PPCoin, all the users are in the same group in order to use GS, and it consists of five phases (i.e., `Initialize`, `Register`, `Transact`, `Chain`, and `Trace`), as shown in Fig. 2. The involved cryptographic schemes are ECDSA, GS, and BE.

1) `Initialize`: This phase first initializes three PNs for the consensus on the first block (named as genesis block) of PPCoin. The system parameter PP generated by `ESetup`, `GSetup`, `BSetup`, and `BSVG` in different cryptographic schemes is recorded in the genesis block. Then, the procedure of mining coins is completely executed once, that is, the world state of RC's account address is initialized with MAX coins (where the balance is encrypted under BE using the public keys of PNs and RC).
2) `Register`: Assuming that a new participant (e.g., user, manager, or PN) joins PPCoin, it must first register in the organization of RC. That is, the user (or manager, or PN) retrieves system parameters (from the genesis block of PPCoin) to generate an account address and private key (apk_u, ask_u) via `EKG`, and sends its account address apk_u, real identity rid_u together with some auxiliary information aux_u to RC for requesting keys of GS and BE. Note that, to obtain an initial balance, the requestor should pay the RC some money as the exchange through other payment methods.
   After receipting a registration request (i.e., $(apk_u, rid_u, aux_u, \sigma_u)$ with a payment, where $\sigma_u = ESign(ask_u, apk_u||rid_u||aux_u))$ from the user, manager, or PN, the RC first verifies this request via `EVerify` and transfers an equal number of coins (corresponding to the payment) to the apk_u (if the request is valid). Note that the specific transfer process can be referred to the next phase. Then, it computes some keys, namely, a group private key gsk_u of GS via `GEnroll` and a public / private key pair (bpk_u, bsk_u) of BE via `BPPKG` and `BKG`, and returns these materials to the requestor (i.e., user, manager, or PN). Note that, the RC maintains a list of real identity rid_u and a tag of gsk_u for subsequent tracing and returns an extra tracing key gtk of GS if the requestor is the manager.
3) `Transact`: Assuming that there is a user (e.g., A) would like to transfer num coins to the other user (e.g., B), the A first retrieves the $nonce$ and $cBalance$ from its world state. Then, it decrypts the $cBalance$ (using its private key of BE via `BDec`) to obtain its current balance, then further prepares the transaction if the balance is not less than num. Specifically, the transaction includes $(beCiphertext, gsSignature)$, where $beCiphertext$ is the ciphertext of $m = (nonce + 1)||gasLimit||gasPrice||addrB||num||v||r||s$ (under the PNs' public keys of BE via `BEnc`), and $gsSignature$ is the GS of $beCiphertext$ using A's group private key via `GSign`. This transaction will be broadcasted into the blockchain network and becomes waiting for the verification and chaining by the PNs.
4) `Chain`: The validating node verifies the transaction as follows. It uses the group public key gpk to invoke `GVerify` for verifying whether the GS (i.e., $gsSignature$) of transaction is validate or not. If not, it discards this transaction; otherwise, it uses its private key of BE to decrypt the ciphertext (i.e., $beCiphertext$) via `BDec` and further checks the validation of ECDSA signature via `EVerify`. This transaction is legitimate, if and only if the ECDSA signature is correct, and the num is not more than the balance in the world state,[8] as well as the nonce is correctly increased by one comparing to that in the world state.
   Those transactions will be sent to recording nodes for being chained into the ledger. Here, the PBFT consensus protocol is adopted to chain transactions. That is, the current leader in this protocol collects enough legitimate transactions for a pending block. Then a consensus of this block will be achieved if and only if not less than two-third of recording nodes approve it. The leader will chain the approved block into the ledger, and the world states of involved user's account addresses will be updated. That is, the nonce of A will be increased by one, and the balance of A will be decreased by num, and correspondingly B's balance will be increased by num. Notably, the balance in world state is updated under the encryption of BE using the user's and PNs' public keys.

[8]Validate nodes can decrypt the ciphertext of balance in the world state by using its private key of BE.

5) `Trace`: This phase is executed by the manager to trace a transaction back to the actual user if it detects an illegal transaction. Specifically, it uses its tracing key gtk to invoke `GTrace` for getting the private information of issuer, and finds the real register information rid with the help of RC. Moreover, the manager could further decrypt the transaction amount by cooperating with PNs.

V. Evaluation and Discussion

In this section, we will discuss the security and privacy of our PPChain, and also give the implementation with performance analysis to show its utility. Note that the applications (e.g., PPCoin) inherit these properties of PPChain, for they are all constructed on the basic of PPChain.

A. Security and Privacy Analysis

Theorem 1: The proposed PPChain satisfies the *anonymity*, and *traceability* if the adopted GS scheme is anonymous and traceable.

Proof: The *traceability* is obvious, which is guaranteed by the `GTrace` in our adopted GS. Here, we mainly prove the anonymity of our PPChain. Supposing that there exists a $\mathcal{PPT}$ adversary $\mathcal{D}$ that could distinguish two transactions generated by different identities (e.g., rid_0 and rid_1) with a nonnegligible advantage ϵ, then we can construct another $\mathcal{PPT}$ adversary $\mathcal{B}$ to break the anonymity of GS. Given a enroll oracle $gsk \leftarrow \mathcal{O}_{GEnroll}(\cdot)$, a GS oracle $gsSignature \leftarrow \mathcal{O}_{GSign}(m,\cdot)$, a tracing oracle $rid \leftarrow \mathcal{O}_{GTrace}(gsSignature,\cdot)$, and a group public key gpk (according to the security model of GS schemes such as [35]), the $\mathcal{B}$ simulates the following oracles queried by $\mathcal{D}$.

1) $\mathcal{O}_r(i)$: Given a query number i, $\mathcal{B}$ invokes `EKG` to generate an address and private key (apk_i, ask_i), and initializes some coins in apk_i. Then, it queries the $\mathcal{O}_{GEnroll}(\cdot)$ to obtain a group private key gsk_i of GS and invokes `BPPKG` and `BKG` to compute a public/private key pair (bpk_i, bsk_i) of BE. Finally, it adds i and $(apk_i, ask_i, gsk_i, bpk_i, bsk_i)$ into the list $\mathcal{L}$ locally and returns (apk_i, bpk_i).
2) $\mathcal{O}_{tt}(num, apk_s, apk_r)$: Given a transaction value num, sender's account apk_s and receiver's account apk_r, $\mathcal{B}$ first uses the account private key ask_s to invoke `ESign` to obtain transaction information $m = (nonce + 1)||gasLimit||gasPrice||addrB||num||v||r||s$. Then, it computes the BE ciphertext $beCiphertext$ upon permission nodes' public key set **bpk** via `BEnc`. Next, it queries the oracle $\mathcal{O}_{GSign}(beCiphertext,\cdot)$ or directly invokes `GSign` using the enrolled group private key gsk_i to obtain the GS $gsSignature$. Finally, it broadcasts tx to blockchain network for being chained by PNs.
3) $\mathcal{O}_c(i)$: Given a query number $i \in \mathcal{L}$, it retrieves the list $\mathcal{L}$ to return $(apk_i, ask_i, gsk_i, bpk_i, bsk_i)$.
4) $\mathcal{O}_{te}(tx)$: Given a transaction tx, it first parses tx as $(beCiphertext, gsSignature)$ and queries the oracle $\mathcal{O}_{GTrace}(gsSignature,\cdot)$ to return the signer's real identity.

After querying the above oracles with enough times, the $\mathcal{D}$ launches the challenge to $\mathcal{B}$ with preparing two untraced transactions tx_0 and tx_1. The $\mathcal{B}$ first parses tx_0 and tx_1 as $(beCiphertext_0, gsSignature_0)$ and $(beCiphertext_1, gsSignature_1)$, respectively, then transmits $gsSignature_0$ and $gsSignature_1$ to the challenger of GS scheme. Thus, the $\mathcal{B}$ will obtain a challenge rid_b from the challenger of GS scheme, and sends rid_b to get the guess b' from $\mathcal{D}$. Finally, $\mathcal{B}$ uses the b' as its answer to the challenger of GS scheme.

This represents that $\mathcal{B}$ can break the anonymity of GS with ϵ, which contradicts that our adopted GS scheme is anonymous and traceable. Thus, the anonymity of our PPChain has been proved.

Theorem 2: The proposed PPChain satisfies the *confidentiality* if the adopted BE scheme is with confidentiality.

Proof: We assume that there exists a $\mathcal{PPT}$ adversary $\mathcal{A}$ could distinguish two transactions of different values (e.g., num_0 and num_1) with a nonnegligible advantage ϵ, then we can construct another $\mathcal{PPT}$ adversary $\mathcal{B}$ to break the confidentiality of BE scheme. Given a set of public keys bpk, a key generation oracle $\mathcal{O}_{BKG}(bpk,\cdot)$, a decryption oracle $msg \leftarrow \mathcal{O}_{BDec}(bc,\cdot)$, the $\mathcal{B}$ simulates the following oracles queried by $\mathcal{A}$. Note that, in the real security model of BE scheme (e.g., [33]), $\mathcal{B}$ can also access other oracles (e.g., set-secret-value, set-public-key, public-key-replacement), but which are not involved in our proof.

1) $\mathcal{O}_r(i)$: Given a query number i, $\mathcal{B}$ invokes `EKG` to generate an address and private key (apk_i, ask_i), and initializes some coins in apk_i. Then, it invokes the $GEnroll$ to obtain a group private key gsk_i of GS and queries the $\mathcal{O}_{BKG}(bpk,\cdot)$ to compute a public/private key pair (bpk_i, bsk_i) of BE. Finally, it adds i and $(apk_i, ask_i, gsk_i, bpk_i, bsk_i)$ into the list $\mathcal{L}$ locally and returns (apk_i, bpk_i).
2) $\mathcal{O}_{tt}(num, apk_s, apk_r)$: Given a transaction value num, sender's account apk_s and receiver's account apk_r, $\mathcal{B}$ first uses the account private key ask_s to invoke `ESign` for obtaining transaction information $m = nonce + 1)||gasLimit||gasPrice||addrB||num||v||r||s$. Then, it computes the BE ciphertext $beCiphertext$ upon its holding public key set **bpk** via `BEnc`, meaning that the **bpk** is regarded as the public key set of PNs. Next, it invokes the `GSign` using the enrolled group private key gsk_i to obtain the GS $gsSignature$. Finally, it broadcasts tx to blockchain network for being chained by PNs.
3) $\mathcal{O}_c(i)$: Given a query number $i \in \mathcal{L}$, it retrieves the list $\mathcal{L}$ to return $(apk_i, ask_i, gsk_i, bpk_i, bsk_i)$.
4) $\mathcal{O}_{te}(tx)$: Given a transaction tx, it first parses tx as $(beCiphertext, gsSignature)$ and queries the oracle $\mathcal{O}_{BDec}(beCiphertext,\cdot)$ to return the original transaction information.

After querying the above oracles with enough times, the $\mathcal{A}$ launches the challenge to $\mathcal{B}$ with preparing two transferred values num_0, num_1 and two uncorrupted accounts apk_s, apk_r. The $\mathcal{B}$ uses the account private key ask_s to obtain two transaction information m_0 and m_1, where $m_i = (nonce + 1) + gasLimit||gasPrice||apk_r||num_i||v||r||s$ for $i = 0, 1$. Then, the $\mathcal{B}$ will get a challenge $beCiphertext_b$ and signs it using the group private key gsk_s via `GSign` to obtain $tx_b = (beCiphertext_b, gsSignature_b)$. Next, the $\mathcal{B}$ sends tx_b to $\mathcal{A}$ and obtains the $\mathcal{A}$'s guess b'. Finally, $\mathcal{B}$ uses the b' as its answer to the challenger of BE scheme.

This means that $\mathcal{B}$ can break the confidentiality of BE with ϵ, which contradicts that our adopted BE scheme is with confidentiality. Thus, the confidentiality of our PPChain has been proved.

In addition to owning the above-proved properties, our PPChain can also resist against some possible attacks. While the

security of PPChain somewhat draws on the inherent security features of Ethereum, there also exist different solutions to these attacks.

1) *Interception and modification attack*: Malicious network attackers may make undiscovered modifications on transactions (e.g., modifying transferred value, account addresses of sender and receiver, and BE ciphertext). In this case, the modified transaction will be invalid for the unforgeability of signatures (i.e., ECDSA and GS) and data nonmalleability of BE.
2) *Double-spending attack*: Some profit-driven users may spend the same single coin more than once. This case can be prevented in PPChain for using the same globally accessible nonce as that in Ethereum, and the PNs will reject the latter transaction with the same nonce.
3) *Birthday collision attack*: There probably exist two blocks are generated at the same time, which will lead to disputes between two sub-blockchains. In PPChain, some permissioned (honest) nodes maintain the ledger according to the PBFT consensus protocol. Thus, the "fork" situation will not appear, which effectively avoids this attack.
4) *Hijacking attack*: To prevent transactions from being hijacking by attackers, the PPChain uses asymmetric cryptography (i.e., ECDSA and GS signatures) to sign transactions with corresponding private keys. Attacker (even a PN) cannot add changes to transaction without breaking these signatures.
5) *Network analysis attack*: In blockchain systems, network attackers may analyze the network history of publicly announced transactions to trace the profiles corresponding to addresses. The transaction in PPChain is signed using GS, which provides unlinkability and untraceability of transactions from the view of normal users. Moreover, the use of BE can hide the account address of sender/receiver and transferred value, and hence efficiently withstands the network analysis attack.
6) *33% attack*: A permissioned blockchain system may be controlled by a group of attackers if these attackers hold more than one-third of the computing power. In our PPChain, the ledger is processed and maintained by PNs that are trusted and dynamically authorized and revoked (once some compromise behaviors were discovered). Thus, the probability is negligible for one to control more than half of PNs. Furthermore, the adopted PBFT can still reach a consensus even at most one-third PNs are compromised.
7) *Impersonation attack*: Attacker may impersonate users or manager to issue a valid transaction or obtain tracing key from RC. This is prohibited in PPChain, for the one hand that only registered participants in RC can enjoy its authority. On the other hand, any impersonator disguising as users (without its account private key) will be identified (even it owns a valid group private key) since the validation of transactions will fail, which is guaranteed by the unforgeability of ECDSA.

B. Performance Analysis

The proposed PPChain is a general architecture, and any GS and BE can be integrated into it. Although we instantiate GS and BE as that proposed in [35] and [33], respectively, other GS and BE schemes can also be used in PPChain. Specifically, we implement these instantiations (including ECDSA) in our personal computer based on the pairing-based cryptography library (version 0.5.12).[9] The concrete system configuration is the Ubuntu system (version 10.10, 32 b) with an Inter (R) Core (TM) i7-6700 CPU of 3.40 GHz and 2 GB RAM, and the used pairing is a type A pairing (with setting the length of group order as 256 and the order of the base field as 512 b long). We tested 1000 times to obtain average time costs of each algorithm in ECDSA, GS, and BE schemes (as shown in Table I). For the subsequent scalability evaluation, we changed the number of receivers from 4 to 32 to test the time costs of BE algorithms.

TABLE I
TIME COST OF CRYPTOGRAPHIC ALGORITHMS (SECOND)

ECDSA				
Algorithm	ESetup	EKG	ESign	EVerify
Time cost	0.006522	0.008112	0.006388	0.009696
BE				
Algorithm	BSetup	BKG	BENC	BDEC
4 receivers	0.010798	0.04465	0.04736	0.004931
8 receivers	0.01097	0.097319	0.08581	0.004799
12 receivers	0.010646	0.130963	0.121646	0.004723
16 receivers	0.01088	0.184511	0.156634	0.004731
20 receivers	0.011033	0.227222	0.195305	0.004771
24 receivers	0.010717	0.267662	0.231441	0.004729
28 receivers	0.010593	0.307456	0.268579	0.00466
32 receivers	0.010839	0.361066	0.309156	0.004719
GS				
Algorithm	GSetup	GEnroll	GSign	GVerify
Time cost	0.0295	0.020939	0.023034	0.02641

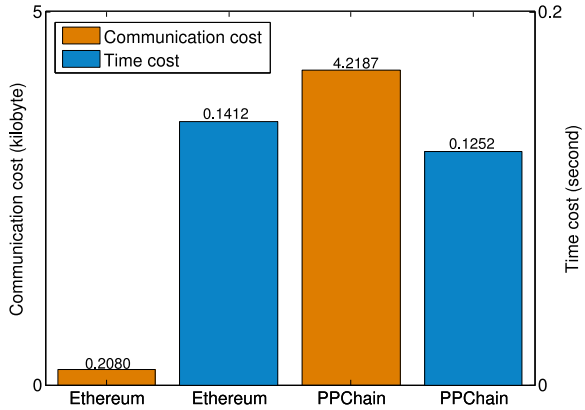


Fig. 3. Evaluation of communication and time cost.

On basic of the above implementation, we first compare PPChain with Ethereum in terms of communication and time costs. Here, we assume that the average block time using PoW consensus is about 15 s and process around eight transactions per second, and that using PBFT consensus can run 1000 transactions per second with latency less than 1 s. This a realistic hypothesis from the view of widely used Ethereum[10] and Hyperledger [6]. Then, we consider combing this consensus time into the time of involved cryptographic primitives (i.e., ECDSA in Ethereum, and ECDSA, GS, BE in PPChain). The number of PNs here is 4 and we obtain the comparative result (see Fig. 3). One can find that the time cost of PPChain is less with that of Ethereum, but the communication cost of PPChain is much

[9]http://crypto.stanford.edu/pbc/
[10]https://www.etherchain.org/

TABLE II
COMPUTATION COST (S) AND COMMUNICATION COST (BYTES)

Items	Bitcoin	Ethereum	Hyperledger	Our PPChain
Type of blockchain	Public	Public	Permissioned	Permissioned
Node-Scalability	H	H	L	L
Performance-scalability	L	L	H	H
Transaction fee	Y	Y	N	N
Mining reward	Y	Y	N	N
Consensus protocol	PoW	PoW	PBFT	PBFT
Privacy of transaction data	N	N	Y	Y
Anonymity	Pseu	Pseu	Pseu	Y
Centralized Regulation	L	M	L	High
Native currency	Y, bitcoin	Y, ether	N	Y, PPCoin
Concurrency	L	L	H	M
Scripting	Limited possibility	High possibility	High possibility	High possibility

* Note that, as the property of centralized regulation, Bitcoin provides decentralized decision making by community/miners, Ethereum is decided by core developer group, Hyperledger is an open-governance model, and PPChain is a manager regulation model. In addition, considering the property of scripting, both Ethereum and PPChain propose Turing-complete virtual machine, high-level language support (i.e., Solidity), Hyperledger also owns Turing-complete scripting of chaincode (i.e., high-level Go-language), but Bitcoin only provides stack-based scripting.

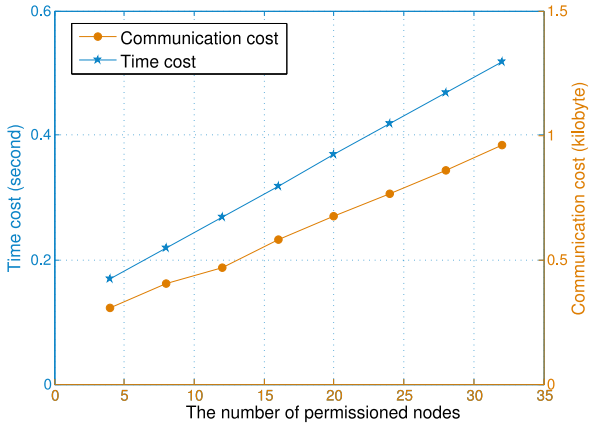


Fig. 4. Scalability of PPChain.

higher than that of Ethereum. This high communication cost is mainly due to the extra GS and BE involved in PPChain, which can be decreased by choosing those GSs and BEs with shorter signature or ciphertext length.

Moreover, to test the scalability of PPChain in terms of cryptographic materials, we vary the number of PNs from 4 to 32. As shown in Fig. 4, both the communication and time costs of PPChain increase linearly as the number of PNs increases. Nevertheless, even the number of PNs reaches to 32, the transaction size is still less than 1 kB, and the corresponding time cost is about 0.5 s. These are acceptable for the scalability of PBFT consensus tested up to 20 nodes in [36] and [37].

Finally, we give a high-level comparison between PPChain and other popular chains (i.e., Bitcoin, Ethereum, Hyperledger) from the perspective of some important blockchain properties. These properties include type of blockchain, scalability of node (i.e., the number of consensus nodes) and performance (i.e., latency, throughput and power consumption), transaction fee, mining reward, consensus protocol, privacy of transaction data (including transferred value and additional data), anonymity (protecting identity of transaction), centralized regulation (tracing real identity of malicious transaction), native currency, concurrency (executing transactions concurrently) and, last but not least, scripting for functionalities of blockchain. Here, we use "Y" and "N" to represent the property offered or not, and abbreviate "High," "Medium," and "Low" as "H," "M," and "L" to denote the level of property, as well as the abbreviation "Pseu" refers to the pseudonymity. From the comparative results in Table II, PPChain owns these advantages, that is, providing enhancing privacy and security protection and also the functionality of regulation, avoiding the transaction fee and mining reward, owning the high-performance scalability, and high possibility of scripting (i.e., supporting solidity). However, like Hyperledger, the PPChain is also faced with the low node-scalability, which is due to the use of PBFT consensus protocol. This has determined the applications of PPChain are those permissioned paradigms requiring the above desirable features.

VI. POTENTIAL APPLICATIONS FOR PPCHAIN

Due to the underlying anonymity, confidentiality and regulation features, the PPChain can be deployed in applications that require security, privacy and reliability. We remark that the function of smart contract in PPChain inherits that of Ethereum, but storage in PPChain is encrypted by BE. This further ensures the security of deploying applications in PPChain. Here, we will introduce two potential applications for PPChain, but the utility of PPChain is not limited to these applications.

A. Food Supply Chain

Building food supply chain can reduce the occurrence of food safety events, and blockchain-based solution can resolve the existing problems in centralized one (i.e., monopolistic, asymmetric, and opaque). In the existing blockchain-based supply chain systems (e.g., [38]–[40]), RFID technology is adopted to collect, circulate and share the related data such as production, processing, and warehousing. For further guaranteeing the reliability and authenticity of this data, they are required to be submitted into the blockchain. This system can cover each enterprise (even compulsory food safety and quality supervision inspection centers) in the whole food supply chain, and hence enables the functionalities including information identification, inquiry, tracking, monitoring and tracing.

However, the transparency of blockchain (e.g., Bitcoin and Ethereum) will cause a serious data privacy leakage by directly storing original data into the chain. For example, by monitoring the data submitted into the blockchain, an enterprise may learn

about kernel information such as output, bids, and mode of transport of its contender. This becomes a threat to the commercial interests of enterprises.

Using PPChain, a privacy-preserving and secure supply chain system could be trivially built. First, a smart contract with the function of mapping transaction identity to the phase of food supply chain. Then, after the data acquisition using RFID (e.g., date of manufacture), each enterprise submits its data into the PPChain by embedding it into a transaction, and builds the connection between the phase and transaction identity in the smart contract. One can retrieve the information of each phase in the whole food supply chain, correspondingly only the PNs can decrypt the data and reply. For the privacy protection mentioned above in PPChain, anyone (except the PNs and manager) cannot obtain original data through the transaction chained in PPChain. Moreover, the traceability of PPChain can also resist malicious data submission and hence guarantees the tracing of a food supply chain system.

B. Sealed-Bid Auctions

Auctions refer to using Ethereum to sell tokens and resources such as Ehereum name service (ENS).[11] In the auction of ENS, someone first initiates an auction on a name and waiting for other user (called as bidder) to bid. Then, bidders get three days to provide their bids, and next each bidder reveals its bid in another two days. The amount of bid will be forfeited if the bidder does not reveal it in time. Finally, the highest bidder obtains the ENS with paying the value of the second highest bid. In existing solutions to ENS auctions, a bidder needs to make two application programming interface (API) calls for its bid. One is the creation of a sealed bid, that is, hashing its bid value with a random secret, and the other is submission of its committed bid with paying a certain amount of ether. This solution requires that a bidder pays more ether than its bid value for hiding the real value of the latter.

This nevertheless provides no good protection of bid privacy for two aspects [41]. For one, the deposited ether discloses some information of the real bid value (i.e., its upper bound). For the other, a very large number of bid value is required for enhancing privacy of bid value. However, this may be not accepted by the bidder, for this amount will be locked until the end of the bidding phase (if successful), or in the worst case until the completion of the whole bid procedure.

PPChain provides a simple way to solve the above issues. That is, the bidder also makes the first API call to generate the sealed bid, but in the second API call, it directly submits the committed bid with paying the equal ether to bid value. For the transaction detail is hided in PPChain, any bidder cannot obtain others' bid values. Then, after all the bidders reveal their bid values, the PNs can decide that each revealed value is identical to the deposit, and then confirm the final winner. One may argue that the PNs can learn details of bidding, which could be mitigated by requiring that a bidder also pay more PPCoins than its bid value.

VII. Conclusion

We introduced a novel permissioned blockchain architecture, seeking to strike a balance between achieving anonymity and having regulation, achieving both transparency and confidentiality. The proposed PPChain employs a modified transaction design (based on that of Ethereum) by integrating with both GS and BE, but eliminates the need for transaction fee and mining reward. Moreover, the PBFT consensus protocol with a validate-record separation mechanism is adopted in the PPChain to improve throughput. Our security analysis and performance analysis, together with potential applications (i.e., PPCoin, food supply chain, and sealed-bid auction), demonstrated the utility of the PPChain.

Future research includes improving the design of GS and BE to achieve better efficiency, and implementing the PPChain in a real-world context.

[11] https://ens.domains/

References

[1] E. Portmann, "Rezension 'Blockchain: Blueprint for a new economy'," *HMD - Praxis der Wirtschaftsinform.*, vol. 55, no. 6, pp. 1362–1364, 2018.

[2] S. Underwood, "Blockchain beyond Bitcoin," *Commun. ACM*, vol. 59, no. 11, pp. 15–17, 2016.

[3] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.

[4] C. Lin, D. He, X. Huang, K. R. Choo, and A. V. Vasilakos, "BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *J. Netw. Comput. Appl.*, vol. 116, pp. 42–52, 2018.

[5] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[6] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.* Porto, Portugal, Apr. 23–26, 2018, pp. 30:1–30:15.

[7] D. B. Rawat, V. Chaudhary, and R. Doku, "Blockchain: Emerging applications and use cases," 2019, *arXiv:1904.12247*.

[8] S. K. Dwivedi, R. Amin, and S. Vollala, "Blockchain based secured information sharing protocol in supply chain management system with key distribution mechanism," *J. Inf. Secur. Appl.*, vol. 54, 2020, Art. no. 102554.

[9] C. Lin, D. He, N. Kumar, X. Huang, P. Vijayakumar, and K. R. Choo, "Homechain: A blockchain-based secure mutual authentication system for smart homes," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 818–829, Feb. 2020.

[10] C. Lin, D. He, X. Huang, X. Xie, and K. R. Choo, "Blockchain-based system for secure outsourcing of bilinear pairings," *Inf. Sci.*, vol. 527, pp. 590–601, 2020.

[11] E. Ben-Sasson *et al.*, "Zerocash: Decentralized anonymous payments from Bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 18–21, 2014, pp. 459–474.

[12] S. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero," in *Proc. 22nd Eur. Symp. Res. Comput. Secur.*, Oslo, Norway, Sep. 11–15, 2017, vol. 10493, pp. 456–474.

[13] G. Fuchsbauer, M. Orrù, and Y. Seurin, "Aggregate cash systems: A cryptographic investigation of Mimblewimble," in *Proc. Adv. Cryptol.—38th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Darmstadt, Germany, May 19–23, 2019, vol. 11476, pp. 657–689.

[14] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "QuisQuis: A new design for anonymous cryptocurrencies," in *Proc. 2019 ASIACRYPT Conf.*, Kobe, Japan, vol. 1, Dec. 8–12, 2019, pp. 649–678.

[15] Y. Li, G. Yang, W. Susilo, Y. Yu, M. H. Au, and D. Liu, "Traceable Monero: Anonymous cryptocurrency with enhanced accountability," *IEEE Trans. Depend. Secur. Comput.*, vol. 18, no. 2, pp. 679–691, Mar.-Apr. 2021.

[16] Y. Wu, H. Fan, X. Wang, and G. Zou, "A regulated digital currency," *Sci. China Inf. Sci.*, vol. 62, no. 3, pp. 32109:1–32109:12, 2019.

[17] S. Meiklejohn *et al.*, "A fistful of Bitcoins: Characterizing payments among men with no names," *Commun. ACM*, vol. 59, no. 4, pp. 86–93, 2016.

[18] G. Maxwell, "Coinswap: Transaction graph disjoint trustless trading," Oct. 2013.

[19] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for Bitcoin with accountable mixes," in *Proc. 18th Int. Conf. Financial Cryptography Data Secur.*, Christ Church, Barbados, Mar. 3–7, 2014, vol. 8437, pp. 486–504.

[20] E. Duffiel and D. Diaz, "Dash: A privacycentric cryptocurrency," 2015. [Online]. Available: https://www.dash.org/

[21] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for Bitcoin," in *Proc. Int. Workshops Financial Cryptography Data Secur.*, San Juan, Puerto Rico, Jan. 30, 2015, vol. 8976, pp. 112–126.
[22] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for Bitcoin," in *Proc. 19th Eur. Symp. Res. Comput. Secur.*, Wroclaw, Poland, Sep. 7–11, 2014, vol. 8713, pp. 345–364.
[23] G. D. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore, "Sybil-resistant mixing for Bitcoin," in *Proc. 13th Workshop Privacy Electron. Soc.*, Scottsdale, AZ, USA, Nov. 3, 2014, pp. 149–158.
[24] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "Coinparty: Secure multi-party mixing of Bitcoins," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy*, San Antonio, TX, USA, Mar. 2–4, 2015, pp. 75–86.
[25] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of Monero's blockchain," in *Proc. 22nd Eur. Symp. Res. Comput. Secur.*, Oslo, Norway, Sep. 11–15, 2017, vol. 10493, pp. 153–173.
[26] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 19–22, 2013, pp. 397–411.
[27] S. Ma, Y. Deng, D. He, J. Zhang, and X. Xie, "An efficient NIZK scheme for privacy-preserving transactions over account-model blockchain," *IEEE Trans. Depend. Secur. Comput.*, vol. 18, no. 2, pp. 641–651, Mar.-Apr. 2021.
[28] N. Narula, W. Vasquez, and M. Virza, "zkLedger: Privacy-preserving auditing for distributed ledgers," in *Proc. 15th USENIX Symp. Netw. ed Syst. Des. Implementation*, Renton, WA, USA, Apr. 9–11, 2018, pp. 65–80.
[29] C. Lin, D. He, X. Huang, M. K. Khan, and K. R. Choo, "DCAP: A secure and efficient decentralized conditional anonymous payment system based on blockchain," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2440–2452, Jan. 2020.
[30] D. R. Brown, "The exact security of ECDSA, " in, *Advances in Elliptic Curve Cryptography*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[31] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, 2001.
[32] A. Fiat and M. Naor, "Broadcast encryption," in *Proc. 13th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA, Aug. 22–26, 1993, vol. 773, pp. 480–491.
[33] S. H. Islam, M. K. Khan, and A. M. Al-Khouri, "Anonymous and provably secure certificateless multireceiver encryption without bilinear pairing," *Secur. Commun. Netw.*, vol. 8, no. 13, pp. 2214–2231, 2015.
[34] D. Chaum and E. van Heyst, "Group signatures," in *Proc. Workshop Theory Appl. Cryptographic Techn.*, Brighton, U.K., Apr. 8–11, 1991, vol. 547, pp. 257–265.
[35] T. Ho, L. Yen, and C. Tseng, "Simple-yet-efficient construction and revocation of group signatures," *Int. J. Found. Comput. Sci.*, vol. 26, no. 5, pp. 611–624, 2015.
[36] M. Vukolic, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Proc. Int. Workshop Open Problems Netw. Secur.*, Zurich, Switzerland, Oct. 29, 2015, vol. 9591, pp. 112–125.
[37] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, Chicago, IL, USA, May 14–19, 2017, pp. 1085–1100.
[38] F. Tian, "An agri-food supply chain traceability system for China based on RFID & blockchain technology," in *Proc. 13th Int. Conf. Serv. Syst. Serv. Manage.*, 2016, pp. 1–6.
[39] F. Tian, "A supply chain traceability system for food safety based on HACCP, blockchain & internet of things," in *Proc. Int. Conf. Serv. Syst. Serv. Manage.*, 2017, pp. 1–6.
[40] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in agri-food supply chain management: A practical implementation," in *Proc. IoT Vertical Topical Summit Agriculture-Tuscany*, 2018, pp. 1–4.
[41] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *Proc. Int. Conf. Financial Cryptography*, Sabah, Malaysia, Feb. 10–14, 2020, pp. 423–443.

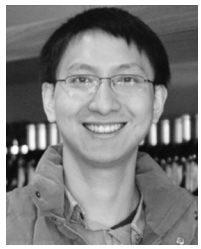
Chao Lin received the Ph.D. degree from the School of Cyber Science and Engineering, Wuhan University, Wuhan, China, in 2020.

He is currently working with the College of Mathematics and Informatics, Fujian Normal University, China. His research interests mainly include applied cryptography and blockchain technology.

Debiao He received the Ph.D. degree in applied mathematics from Wuhan University, Wuhan, China, in 2009.

He is currently a Professor of the School of Cyber Science and Engineering, Wuhan University. His research interests include cryptography and information security, particularly cryptographic protocols. He has authored/coauthored more than 100 research papers in refereed international journals and conferences, such as the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSIC, and *Usenix Security Symposium*.

Dr. He was the recipient of the 2018 IEEE Systems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award. His work has been cited more than 7000 times at Google Scholar. He is in the editorial board of several international journals, such as *Journal of Information Security and Applications*, *Frontiers of Computer Science*, and *Human-Centric Computing and Information Sciences*.



Xinyi Huang received the Ph.D. degree from the University of Wollongong, Wollongong, NSW, Australia, in 2009.

He is currently a Professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, and the Co-Director of Fujian Provincial Key Laboratory of Network Security and Cryptology. His research interests include applied cryptography and network security.

Dr. Huang is an Associate Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. He is on the editorial board of *International Journal of Information Security* (Springer), and was the Program/General Chair or Program Committee Member in more than 80 international conferences.

Xiang Xie received the Ph.D. degree from Chinese Academy of Sciences, Beijing, China, in 2015.

He is currently with the Juzix Technology Co. Ltd., Shenzhen, China. His research interests include public-key cryptography, lattice-based cryptography, block-chain security, and multiparty computation.

Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He is currently the Cloud Technology Endowed Professor with The University of Texas at San Antonio, San Antonio, TX, USA.

Dr. Choo was the recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher), Outstanding Associate Editor of 2018 for IEEE ACCESS, British Computer Society's 2019 Wilkes Award Runner-Up, 2019 EURASIP Journal on Wireless Communications and Networking Best Paper Award, Korea Information Processing Society's Journal of Information Processing Systems Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, IEEE TrustCom 2018 Best Paper Award, and the ESORICS 2015 Best Research Paper Award. He is also a Fellow of the Australian Computer Society, and Co-Chair of IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group.