zkCross: A Novel Architecture for Cross-Chain Privacy-Preserving Auditing

Yihao Guo
Shandong University

Minghui Xu*
Shandong University

Xiuzhen Cheng
Shandong University

Dongxiao Yu
Shandong University

Wangjie Qiu
Beihang University

Gang Qu
University of Maryland

Weibing Wang
Cloud Inspur Information Technology Co., Ltd.

Mingming Song
Cloud Inspur Information Technology Co., Ltd.

Abstract

One of the key areas of focus in blockchain research is how to realize privacy-preserving auditing without sacrificing the system's security and trustworthiness. However, simultaneously achieving auditing and privacy protection, two seemingly contradictory objectives, is challenging because an auditing system would require transparency and accountability which might create privacy and security vulnerabilities. This becomes worse in cross-chain scenarios, where the information silos from multiple chains further complicate the problem. In this paper, we identify three important challenges in cross-chain privacy-preserving auditing, namely Cross-chain Linkability Exposure (CLE), Incompatibility of Privacy and Auditing (IPA), and Full Auditing Inefficiency (FAI). To overcome these challenges, we propose zkCross, which is a novel two-layer cross-chain architecture equipped with three cross-chain protocols to achieve privacy-preserving cross-chain auditing. Among these three protocols, two are privacy-preserving cross-chain protocols for transfer and exchange, respectively; the third one is an efficient cross-chain auditing protocol. These protocols are built on solid cross-chain schemes to guarantee privacy protection and audit efficiency. We implement zkCross on both local and cloud servers and perform comprehensive tests to validate that zkCross is well-suited for processing large-scale privacy-preserving auditing tasks. We evaluate the performance of the proposed protocols in terms of run time, latency, throughput, gas consumption, audit time, and proof size to demonstrate their practicality.

1 Introduction

Cross-chain technology [2] has been considered to be one of the effective ways to enhance communications and interoperability among separate blockchain systems. Users located at different blockchains can realize cross-chain transfer and exchange through cross-chain protocols. Many cross-chain protocols have arisen recently [15, 30, 36, 40], with their primary focuses residing on various aspects such as universality, security, and efficiency, largely neglecting privacy and auditing challenges. This omission may impede the widespread adoption of cross-chain mechanisms and result in significant security vulnerabilities, privacy risks, and even the loss of trust and credibility.

Privacy protection and auditing are two vital requirements in cross-chain operations. As the awareness of the worth of data grows, so does the desire for privacy protection [23, 38]. Nevertheless, privacy considerations in cross-chain need to protect not only the confidentiality of information being transferred across different blockchains but also the interactions among users across multiple chains. Disclosure of the relationship of cross-chain interactions can lead to a number of privacy risks, including the possibility of tracing the transfer of funds between chains or identifying parties involved in a specific transaction. This can potentially undermine the overall security of cross-chain systems [6]. Additionally, auditing plays an important role in cross-chain scenarios. For instance, auditing is needed to ensure the proper functioning of the interactions between two parties that need to collaborate but may potentially have conflicts of interest [22].

However, privacy protection and auditing in blockchain systems are two conflicting goals, as the former demands maintaining data confidentiality and minimizing data exposure, while the latter requires transparency and access to data. Therefore juggling the issues of privacy protection and auditing is difficult, which has long been an important research topic for single blockchains [4, 7, 9, 34, 37]. This problem becomes even more challenging in cross-chain. Specifically, the design premise of a single chain assumes that the auditors, serving as blockchain nodes, can access information within the chain. However, a multi-chain scenario breaks this premise as auditors of one chain cannot obtain all information from another chain. Intuitively, auditors can register accounts on all chains, but this is extremely resource-intensive and impractical for real-world applications, particularly in large-scale cross-chain scenarios. For instance, the ledger sizes of

*Corresponding author

Bitcoin and Ethereum have reached 500 and 700 GB, respectively, and they continue to grow over time [17]. This implies that when auditing Bitcoin and Ethereum, an auditor requires at least terabyte-level storage space, and as the number of blockchains in the network grows, this demand also increases. This poses a significant barrier for auditors. As a result, it is hard for single-chain solutions to be inherited in cross-chain.

This paper aims to make a contribution by first identifying three key challenges faced by cross-chain privacy protection and auditing, then presenting feasible solutions. The first challenge is *the Cross-chain Linkability Exposure* problem (CLE), which enables adversaries to obtain linkability between the two parties involved in cross-chain interactions, thereby exposing privacy. Addressing this issue requires designing corresponding cross-chain privacy-aware protocols for different cross-chain activities. These protocols must not permit the involvement of a third party that could compromise the security of the entire blockchain system. Second, we need to implement an audit protocol based on publicly available information, and this protocol should not contradict the protocol aimed at addressing the CLE problem. Therefore the bottleneck of cross-chain privacy protection with auditing, namely *the Incompatibility of Privacy and Auditing (IPA)*, needs to be overcome. The third challenge is termed *Full Auditing Inefficiency (FAI) problem*, which is caused by the presence of multiple heterogeneous chains that may lead to extremely low auditing efficiency. These challenges are detailed in Section 3.2.

In this paper, we propose zkCross to effectively address these three challenges. Specifically, we first present a two-layer cross-chain architecture, with the lower layer including multiple blockchains that can interact with each other, and the upper layer being a blockchain consisting of auditors. Then, in the pursuit of addressing the CLE problem, we propose two different cross-chain privacy protection protocols based on zero-knowledge proofs for two cross-chain activities (namely cross-chain transfer and cross-chain exchange), which preserve privacy without requiring any trusted third party. Finally, in an effort to address the IPA and FAI problems, we present a cross-chain auditing protocol, which compresses the auditing processes and transaction verification process into a new circuit. Such a design ensures that all auditors do not need to obtain the entire transaction content to guarantee the correctness of auditing tasks, and they can audit a linearly growing number of transactions within a constant time.

zkCross is developed for general blockchain systems with cross-chain interoperability requirements. That is, it is applicable to both permissionless and permissioned blockchains (introduced in Section 2). For convenience, we highlight our contributions as follows:

1. zkCross is the first cross-chain scheme that considers both privacy protection and auditing for cross-chain activities. To the best of our knowledge, solutions that simultaneously consider both privacy protection and auditing only exist in single-chain systems while it is hardly possible to adopt them for cross-chain operations.

2. A two-layer cross-chain architecture is designed to support the realization of simultaneous privacy protection and auditing. This architecture is auditor-friendly. It allows the decoupling of itself with the cross-chain interaction protocols, making the design of the latter more flexible.

3. Two cross-chain privacy-preserving protocols are proposed in this paper, which can guarantee the unlinkability of the interacting parties, ensure the atomicity of cross-chain interactions, and further solve the CLE problem without introducing any trusted third party.

4. We design a cross-chain auditing protocol, that can effectively address the challenges of IPA and FAI. This protocol can enhance the auditing efficiency of the upper layer, without causing conflicts with addressing the CLE issue.

5. To verify the performance of zkCross, we conduct extensive experiments over a 200-node blockchain network, and the results validate the effectiveness and efficiency of our design.

The remainder of this paper is structured as follows. In Section 2, we provide the background of our work. Section 3 provides an overview of the most relevant literature and proposes the three cross-chain challenges. The models of our proposed scheme, along with a brief introduction to the necessary preliminary knowledge, are described in Section 4. The technical details of our zkCross scheme and the related security analysis are presented in Section 5 and Section 6, respectively. The performance of zkCross is reported in Section 7. Finally, Section 8 offers conclusive remarks and directions for future research.

2 Background

This section provides the necessary background to understand our work, including blockchain and cross-chain activities.

2.1 Blockchain

Blockchain [32] is a shared, distributed ledger that records all transactions and is maintained by peer nodes in a decentralized network. Each transaction is stored in a block and linked together in a chronological chain, hence the name blockchain. One of the key features of blockchain is its immutability, meaning once a transaction is recorded, it cannot be altered. Immutability ensures the integrity of the data stored on the blockchain, making it secure and resistant to unauthorized modifications. Additionally, blockchain eliminates the need for intermediaries in transactions, solving the problem of a single point of failure. Transparency in blockchain denotes the ability to access all transactions. This is in contrast to traditional banks and service providers, which do not grant users access to their complete ledgers. Originally developed for Bitcoin [32], blockchain has since evolved to be applied in various industries beyond cryptocurrency, such as finance, the Internet of Things, smart grids, and supply chain management [45]. To accommodate various scenarios, blockchain has gradually diverged into two types: permissionless and permissioned blockchains [38]. Permissionless blockchains are decentralized networks where anyone can join, participate, and validate transactions without needing approval. Permissioned blockchains are centralized or semi-centralized networks where access and participation are restricted to approved entities.

2.2 Cross-chain Activities

Cross-chain technology facilitates the interaction between two independent blockchains [2]. Depending on the type of cross-chain activities, cross-chain interactions can be divided into cross-chain transfers and cross-chain exchanges. Cross-chain transfers refer to the movement of assets between two blockchains, which involve the process of burning assets on one blockchain and minting them on another blockchain. Cross-chain exchanges, conceptually, involve a two-way process of cross-chain transfers. They refer to the swapping of assets between two distinct blockchains. Both participating blockchains individually lock the assets intended for exchange and subsequently unlock them to receive the corresponding assets from the other blockchain.

While the concept of these two activities may appear similar, the specific technical and operational aspects of cross-chain asset transfer and exchange protocols are different and require distinct approaches and solutions. For instance, various cross-chain exchange protocols, such as HTLC and the works in [6, 18, 30], do not support asset transfer. While other schemes such as the notary scheme and relay chains (introduced in Section 3), have the capability to facilitate both cross-chain asset transfer and exchange, albeit often requiring the involvement of third-party entities possessing significant authority, thereby introducing centralized security risks.

3 Related Work and Motivations

In this section, we first briefly introduce the most related work for cross-chain interactions, then present the motivations and implications of our design by detailing the three cross-chain challenges that have largely been overlooked by the related research.

3.1 Related Work

State-of-the-art cross-chain schemes can be divided into two categories: either chain-based or bridge-based. Chain-based schemes, such as side chains [28] and hashed timelock contract (HTLC) [31], refer to those that rely on the inherent mechanisms of a chain, without the requirement for additional entities. A side chain is a separate blockchain that operates in conjunction with a parent blockchain. ZeroCross [21] is a privacy-preserving side chain solution based on the Monero platform, and Zedoo [10], which implements auditing based on the zk-SNARK technology. Baldimtsi *et al.* [1] proposed a scheme to connect anonymous chains. HTLC utilizes a combination of a hash lock and a time lock to support the cross-chain exchange activity XCLAIM [41] alleviates some of the stringent assumptions inherent in the original HTLC framework, such as the necessity for the concurrent online presence; Deshpande *et al.* [6] added privacy protection features to HTLC based on the Schnorr signatures; MAD-HTLC [31] effectively counters the incentive manipulation attack utilizing a blockchain-based incentive mechanism; Cross-Channel [15] increases the throughput of a cross-chain system based on HTLC; Thyagarajan *et al.* [30] employed verifiable timed signatures instead of time locks.

Bridge-based solutions, such as notary schemes [40] and relay chains [2], need the implementation of a third component to facilitate the interactions among chains. In the context of a notary scheme, a trusted third-party entity, referred to as a notary, performs the duties of validating the authenticity of cross-chain transactions. The design of a notary scheme is straightforward; however, it poses a potential single point of failure risk. Yin *et al.* [40] adopted secure hardware and cryptographic techniques to enhance the security of notary platforms. As for a relay chain scheme, a dedicated blockchain network, referred to as the relay chain, serves the purpose of enabling the transfer of assets between participating chains. zkBridge [36] proposed deVirgo to enhance the verification efficiency of cross-chain bridges. Wang *et al.* [33] proposed the concept of governing the chain by chain while BeDCV [44] employs a supervision chain for decentralized auditing, utilizing technologies such as homomorphic encryption. Note that, bridge-based schemes introduce trusted mechanisms such as notaries and relay chains, which enable the simultaneous consideration of both cross-chain transfers and cross-chain exchanges. Moreover, the bridge-based scheme is a type of star-shaped architecture, making it suitable for auditing tasks. However, it carries the risk of a single point of failure.

3.2 Challenges of Cross-Chain Privacy Protection & Auditing

Despite the availability of several well-conceived cross-chain solutions, significant challenges persist in the areas of cross-chain privacy preservation and auditing, including the Cross-chain Linkability Exposure problem (CLE), the Incompatibility issue of Privacy and Auditing (IPA), and the Full Auditing Inefficiency problem (FAI). The details of these three challenges are presented below.

Challenge 1: CLE. Some solutions have been proposed to address cross-chain privacy concerns. Specifically, the schemes in [39, 43] are highly tied to the application scenarios. The privacy protection in [1, 21] benefits from its underlying blockchain platform, such as Monero and Zcash, which makes it incompatible with other blockchain systems. The mechanism in [6] provides a solution for cross-chain exchange but requires both parties to trust the same secret as a prerequisite, which creates challenges for two individuals who lack trust in one another to reach an agreement. Based on the above analysis, it is evident that the current research on cross-chain privacy protection is insufficient, and there also exist concerns regarding their generality and security risks. Additionally, existing works overlook the significant differences between cross-chain transfer and cross-chain exchange, attempting to use a single protocol to support both activities, which needs third-party entities and thereby poses security risks (such as the bridge-based scheme). One can see that the difficulty in addressing the CLE problem lies in comprehending and differentiating the activities and characteristics of blockchain interactions while using more adaptable techniques to obscure the relations between the participants. More importantly, the entire process cannot introduce third parties to disrupt the security of the system.

Challenge 2: IPA. The demands for both privacy protection and auditing often exist concurrently in a system and these two requirements have conflicting ultimate goals. To the best of our knowledge, no existing cross-chain work can solve this conflict. Current cross-chain privacy-preserving solutions [1, 6, 21, 39, 43] are not initially designed with auditing support in mind. Their primary focus is directed towards simultaneously minimizing the disclosure of information and ensuring the integrity of the protocols, disregarding considerations for information verifiability. Moreover, considering that in permissioned blockchains, auditors are external to a blockchain, they may not have access to the transaction information. For example, in a multi-chain system comprising several companies, internal nodes engage in frequent interactions, while third-party entities, such as banks and insurance companies, assume the role of auditors. The privacy of internal data, including receipts and employee information, cannot be disclosed to external entities. However, this restriction prevents banks from conducting audits, consequently impeding their ability to perform operations such as lending. To effectively address the above issues, it becomes crucial to enable the validation of private information by leveraging publicly accessible data.

Challenge 3: FAI. Current auditing works can be further broken down into two distinct subcategories: incentive-based and relay-based. The former, such as the fisherman in Polkadot [35], utilizes an incentive mechanism in which auditors selectively verify a subset of transactions to get the corresponding reward. The latter employs a relay mechanism, such as an entity or a network, to facilitate the thorough examination of all transactions in order to mitigate the risks of malicious activities [33, 44]. In general, a relay-based auditing scheme offers better security as it audits all transactions (full auditing), however, it may be less efficient than an incentive-based audit scheme. Specifically, assuming k blockchains and an average of n transactions processed per chain in a unit of time, the workload responsibility of each audit node in a relay-based scheme can be estimated as $O(\mathsf{k} \times \mathsf{n})$. This poses a high computational demand for auditors and leads to issues with inefficient verification. More seriously, as the number of blockchains continues to grow, the limitations of both methods in terms of auditing efficiency pose a challenge to their adoption in real-world settings. To address this issue, it is crucial to design an auditor-friendly scheme that guarantees security and efficiency while providing full auditing.

Motivations and Implications. Motivated by the above observations, we propose zkCross consisting of a novel architecture with three protocols. Compared to bridge-based schemes (introduced in Section 3), the architecture in zkCross adopts a tree-shaped structure. This design takes into consideration the requirements of both privacy protection and auditing, mitigating the centralization risks posed by auditing in cross-chain interactions. In the design of privacy-preserving protocols, zkCross utilizes SPV (Simplified Payment Verification) [32] and HTLC (Hashed Timelock Contracts) [31], two fundamental techniques supporting decentralized cross-chain interactions, to facilitate cross-chain transfers and exchanges. zkCross adopts zk-SNARKs, hash functions, and denomination mechanisms to address all factors that compromise the unlinkability, including receiver addresses, preimages, and transaction amounts. For the design of auditing protocols, zkCross leverages the succinctness of zk-SNARKs and offloads complex computational tasks to off-chain, thereby enhancing on-chain auditing efficiency.

4 The Model and Preliminaries

In this section, we first define our network and threat models. Then, we present the zk-Rollup, hashed timelock contracts, and simplified payment verification, which are the basic building blocks of zkCross.

4.1 Models

Network Model. zkCross is a two-layer cross-chain architecture equipped with three protocols. All entities in zkCross refer to real-world users who own blockchain accounts and can engage in off-chain and on-chain operations. Furthermore, each entity can possess multiple accounts, with each account equivalent to a node on the blockchain.

In zkCross, the lower layer has multiple independent blockchains (which we refer to as ordinary chains), where there are cross-chain transactions between a sender ($\mathcal{S}$) and a receiver ($\mathcal{R}$). Each node in an ordinary chain can serve as a committer ($\mathcal{C}_\mathcal{T}$), facilitating the aggregation of chain-related data. We assume that there is at least one honest committer in each ordinary blockchain [36]. The ordinary chain can be either permissionless or permissioned blockchains. In permissionless blockchains, all entities can register accounts and access transactions. In permissioned blockchains, external nodes, such as auditors in the upper layer, cannot access transactions as freely as in permissionless chains. Consequently, we assume that auditors may only gain access to block headers of permissioned blockchains through auxiliary means, such as Gateways [25] or blockchain explorers, to facilitate auditing procedures while preserving transaction privacy.

The upper layer comprises an audit chain, including nodes with two roles: committers and auditors. To upload the collected chain-related data to the audit chain and earn rewards, the lower-layer committer needs to register an account and become an upper-layer committer on the audit chain. Auditors, represented by $\mathcal{A}_\mathcal{D}$, are responsible for auditing the data submitted by upper-layer committers. Based on the incentive mechanism of blockchains, honest upper-layer committers are duly rewarded. The audit chain is a permissionless blockchain because it allows all entities to register committer nodes on the audit chain.

Threat Model. Nodes are running in polynomial time and might exhibit Byzantine behaviors within the blockchain network. However, their voting power cannot exceed predetermined thresholds to maintain the security of the blockchain consensus [15, 30, 31, 36]. For example, in PBFT (Practical Byzantine Fault Tolerance), the number of Byzantine nodes must be below one-third of the total, and in PoW (Proof of Work), malicious nodes cannot control more than 50% of the network's computing power.

Design Goals. zkCross aims to achieve the following goals to answer the challenges in Section 3.2.

- Privacy: Let $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{II}}$ be two accounts involved in a cross-chain transfer or exchange. zkCross aims to ensure the unlinkability (formally defined in Definition 1) between $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{II}}$.

- Efficiency: If a cross-chain network has k ordinary chains, where each chain has an average of m blocks, and each block contains an average of n transactions, then zkCross has an $O(\mathsf{k}\times\mathsf{m})$ complexity in auditing efficiency, comparing to $O(\mathsf{k}\times\mathsf{m}\times\mathsf{n})$ for the traditional full auditing approaches.

Definition 1 (Unlinkability [3, 11, 24]). *An adversary is unable to link the receiver's account from the transactions initiated by the sender, or conversely.*

In the rest of this section, we will introduce the three key technologies that we adopt in zkCross to help achieve these goals.

4.2 zk-Rollup Based on zk-SNARKs

zk-Rollup [38] is a scaling solution for blockchain networks, in which the "zk" stands for "zero-knowledge". zk-SNARK [13] is a type of zero-knowledge proof technology that is widely employed in blockchain. In the following, we provide the definition of zk-Rollup based on zk-SNARK. Note that, depending on a specific use case, alternative zero-knowledge proof schemes such as deVirgo [36], can be selected to replace zk-SNARK in the construction of zk-Rollup.

Definition 2 (zk-Rollup Based on zk-SNARK). *zk-SNARK requires the implementation of an $\mathbb{F}$-arithmetic circuit Λ, which enforces constraints on the computational relationships between the public input $\vec{\mathsf{x}}$ and the private input (witness) $\vec{\mathsf{w}}$. The complete process of zk-Rollup based on zk-SNARK can be formally expressed as a three-tuple of polynomial time algorithms* $\Pi \overset{def}{=} (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$.

• $\mathsf{crs} \leftarrow \mathtt{Setup}(1^\lambda, \Lambda)$. In zk-SNARK, the algorithm takes a security parameter 1^λ and a circuit Λ as inputs to derive the common reference string (crs), which is comprised of the proving and verification keys $(\mathsf{pk}, \mathsf{vk})$ used in subsequent proof generation and verification processes; while in zk-Rollup, the computational logic embedded within the circuit entails the verification of both the correctness of all transaction signatures and of state transitions resulting from the executed transactions.

• $\pi \leftarrow \mathtt{Prove}(\mathsf{pk}, \vec{\mathsf{x}}, \vec{\mathsf{w}})$. In zk-SNARK, the algorithm takes the proving key pk, the public inputs $\vec{\mathsf{x}}$, and the private inputs $\vec{\mathsf{w}}$, to generate a succinct zero-knowledge proof π; while in the process of implementing zk-Rollup, the public inputs $\vec{\mathsf{x}}$ include more details such as transfer amounts and address information, and the private inputs $\vec{\mathsf{w}}$ consist of the data such as transaction signatures.

• $1/0 \leftarrow \mathtt{Verify}(\mathsf{vk}, \vec{\mathsf{x}}, \pi)$. In both zk-SNARK and zk-Rollup, the algorithm performs a verification process of the proof π using the verification key vk and public inputs $\vec{\mathsf{x}}$. Its output is a binary value, with 1 indicating a successful verification and 0 the opposite.

Note that, the three-tuple of the polynomial time algorithms (Setup, Prove, Verify) represented by Π, can be adapted to various specific application scenarios by adjusting the relevant parameters, i.e., Λ, $\vec{\mathsf{x}}$, and $\vec{\mathsf{w}}$.

4.3 Hashed Timelock Contracts

HTLC (Hashed TimeLock Contract) is a critical technology that powers the Lightning Network. It has broad applications, especially for cross-chain exchanges [31].

Definition 3 (HTLC in Cross-chain Scenarios)**.** *The HTLC protocol necessitates that both parties involved in a transaction, represented by $\mathcal{S}$ and $\mathcal{R}$, possess accounts in each blockchain. For better elaboration, these accounts are denoted as $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{I}}$ for* Chain I*, and $\mathcal{S}^{\mathsf{II}}$ and $\mathcal{R}^{\mathsf{II}}$ for* Chain II*, for $\mathcal{S}$ and $\mathcal{R}$, respectively. The protocol consists of three distinct stages:* Lock*,* Unlock*, and* Refund.

- Lock : $\mathcal{S}^{\mathsf{I}}$ selects a random key, such as a 256-bit integer, as the preimage pre, which is then hashed to produce the corresponding hash value $\mathsf{h}(\mathsf{pre})$. $\mathcal{S}^{\mathsf{I}}$ then employs $\mathsf{h}(\mathsf{pre})$ to lock the asset to be sent to $\mathcal{R}^{\mathsf{I}}$ in the smart contract ξ^{I} of Chain I, and sets a timer T_1. Then, $\mathcal{R}^{\mathsf{II}}$ locks its asset being sent to $\mathcal{S}^{\mathsf{II}}$ in the smart contract ξ^{II} of Chain II using the same $\mathsf{h}(\mathsf{pre})$, and sets a timer T_2 with $\mathsf{T}_1 > \mathsf{T}_2$.
- Unlock : Within the time limit T_2, $\mathcal{S}^{\mathsf{II}}$ presents the preimage pre to the smart contract ξ^{II} in order to unlock the asset sent by $\mathcal{R}^{\mathsf{II}}$. Once $\mathcal{R}^{\mathsf{II}}$ receives the preimage, $\mathcal{R}^{\mathsf{I}}$ provides pre to the smart contract ξ^{I} within the time limit T_1, which unlocks the asset sent by $\mathcal{S}^{\mathsf{I}}$.
- Refund : If $\mathcal{S}^{\mathsf{II}}$ fails to provide pre within the allotted time frame T_2, the locked asset in ξ^{II} would be refunded to $\mathcal{R}^{\mathsf{II}}$. As a result, $\mathcal{R}^{\mathsf{I}}$ cannot unlock the asset it should receive from $\mathcal{S}^{\mathsf{I}}$ since it does not possess the necessary pre information, which only $\mathcal{S}$ possesses. Consequently, in the event of T_1 timing-out, the locked asset would be refunded to $\mathcal{S}^{\mathsf{I}}$.

Note that, certain blockchain systems, including Bitcoin, do not have native support for smart contracts. As an alternative, HTLC can be implemented using other mechanisms such as scripting [32]. For simplicity, this paper uses the term smart contract to refer to a true smart contract or an implementation technique such as scripting when discussing HTLC.

4.4 Simplified Payment Verification

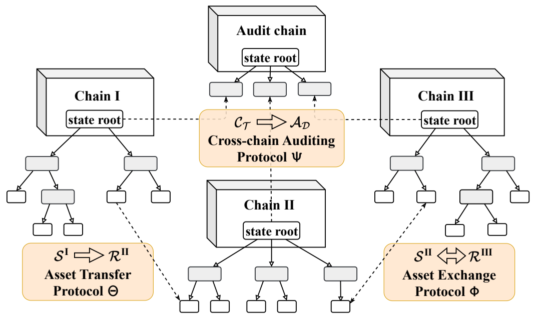
SPV (Simplified Payment Verification) was proposed by Satoshi Nakamoto [32] in 2008 and has been widely adopted by various blockchain systems such as Ethereum since its birth. It aims to enable lightweight clients to participate in a blockchain network without having to store the entire on-chain data by making use of a Merkle proof, a crucial component that allows a light node to verify the inclusion of a transaction in a block without downloading the entire block. Specifically, the Merkle proof of a transaction is the combination of the Merkle path of that transaction and the root node of the entire Merkle tree. The Merkle path of a node is the siblings of all nodes along the path from the node to the root of the Merkle tree.

5 The zkCross

In this section, we first present an overview of zkCross and its advantages, then elaborate on three protocols designed for the purpose of privacy-preserving auditing.

5.1 Overview

zkCross addresses the aforementioned challenges of CLE, IPA, and FAI. Figure 1 depicts its two-layer architecture and three key protocols.



Audit chain
state root
Chain I
state root
Chain III
state root
$\mathcal{C}_{\mathcal{T}} \Rightarrow \mathcal{A}_{\mathcal{D}}$
Cross-chain Auditing Protocol Ψ
Chain II
state root
$\mathcal{S}^{\mathrm{I}} \Rightarrow \mathcal{R}^{\mathrm{II}}$
Asset Transfer Protocol Θ
$\mathcal{S}^{\mathrm{II}} \Leftrightarrow \mathcal{R}^{\mathrm{III}}$
Asset Exchange Protocol Φ

Figure 1: Illustration of zkCross: upper-layer: the audit chain; lower-layer: multiple ordinary chains; and the three protocols shown in the orange boxes.

The lower layer is comprised of multiple ordinary chains, three in Figure 1 marked as Chain I, Chain II, and Chain III, respectively. The upper layer encompasses an audit chain that regulates the activities of the ordinary chains in the lower layer. Within this architecture, each blockchain maintains a state tree, which is essentially a binary Merkle tree. In an ordinary chain, each leaf in its state tree represents the hash result of an account state, encompassing public keys and balances. For the audit chain, its state tree stores the state tree roots of ordinary chains as leaves, establishing hierarchical links between ordinary and audit chains.

zkCross includes two privacy protection protocols (Θ and Φ), tailored for the two distinct cross-chain interaction activities: transfers and exchanges, respectively. Users participating in these protocols are nodes at different ordinary chains. For instance, a sender $\mathcal{S}^{\mathsf{I}}$ in Chain I, can employ protocol Θ to secretly transfer cross-chain assets to a receiver $\mathcal{R}^{\mathsf{II}}$ in Chain II, while a sender $\mathcal{S}^{\mathsf{II}}$ in Chain II can conduct a cross-chain exchange with a receiver $\mathcal{R}^{\mathsf{III}}$ in Chain III through the privacy-preserving protocol Φ. Both Θ and Φ can protect the privacy of the relationship between the interacting parties. Entities in the audit chain primarily consist of auditors and committers. Under the cross-chain auditing protocol Ψ, a committer is responsible for packaging the transactions in an ordinary chain and uploading them to the auditors to complete the auditing process.

The two-layer architecture is capable of supporting the interactions between various entities, such as cross-chain interacting entities and auditing entities, thereby facilitating the design of privacy-preserving cross-chain protocols and efficient auditing protocols. Note that the proposed architecture decouples auditing from cross-chain protocols (such as Θ and Φ) among ordinary chains. This implies that the two-layer architecture is not restricted to privacy-preserving auditing, it can support other protocols developed according to the specific application scenarios. On the other hand, the cross-chain protocols designed in this paper are not restricted to the two-layer architecture only, they are implemented with popular cross-chain technologies, such as SPV and HTLC, and hence can be adopted directly or with modifications for other multiple-chain architectures. The audit content of the auditing protocol can be tailored to specific requirements, encompassing tasks such as verifying transaction amounts within a predefined range and checking if a transaction address is blacklisted. The protocol Ψ can conduct audits without exposing the information of the lower-layer transactions.

In the remainder of this section, we elaborate on the two privacy-preserving cross-chain protocols and the efficient auditing protocol.

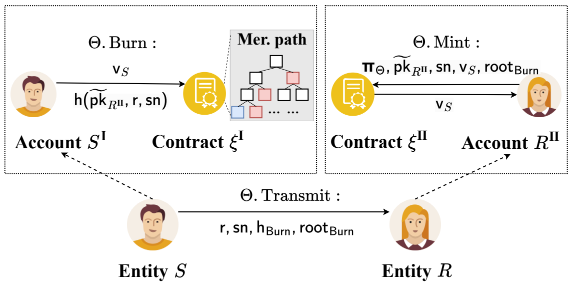
5.2 Privacy-preserving Cross-chain Protocols

In this subsection, we present two different privacy-preserving protocols to address the CLE problem.

5.2.1 An Cross-chain Transfer Protocol with Privacy Preservation

In the process of cross-chain transfer based on SPV, the receiver address and the transfer amount of a transaction are two important factors that can expose the correlation between the parties involved in the interaction. Obviously, one needs to hide the destination address of a transfer. Concealing the transfer amount is necessary too, as the process of a cross-chain transfer involves the burning of funds on one chain and the minting of an equivalent amount on another chain. For example, consider a situation where the public exchange rate between Bitcoin and Ether is 1:10. During a cross-chain transfer, a fund in the form of one Bitcoin is burned on its original chain, and as a result, 10 Ether of equivalent value is minted on the destination Ethereum. Therefore, if the transfer amount is public, adversaries can deduce the correlation between the parties by examining the burning and minting transactions on the two chains. Nevertheless, if the above two privacy-sensitive information are both hidden, they would cause SPV authentication to fail and disrupt the atomicity of the cross-chain interaction.

Based on the above analysis, we design a privacy-preserving cross-chain transfer protocol Θ (shown in Figure 2) to address the privacy concerns arising from the receiver's address and the transfer amount. We choose to hide the receiver's address via zk-SNARK but conceal the transfer amount by setting a fixed denomination [12, 16, 29]. For example, on an Ethereum chain, if the basic denomination is set to 2 Ether when $\mathcal{S}$ intends to transfer 6 Ether to $\mathcal{R}$, $\mathcal{S}$ needs to make separate transfers of 3 2-Ether transfers. Such a design makes use of more transactions to protect the privacy of the transfer value. The whole protocol Θ is presented as follows.



Θ. Burn :
v_S
$\mathsf{h}(\widetilde{\mathsf{pk}}_{R^{\mathrm{II}}}, \mathsf{r}, \mathsf{sn})$
Mer. path
Account S^{I}
Contract ξ^{I}
Θ. Mint :
$\pi_\Theta, \widetilde{\mathsf{pk}}_{R^{\mathrm{II}}}, \mathsf{sn}, \mathsf{v}_S, \mathsf{root}_{\mathsf{Burn}}$
v_S
Contract ξ^{II}
Account R^{II}
Θ. Transmit :
$\mathsf{r}, \mathsf{sn}, \mathsf{h}_{\mathsf{Burn}}, \mathsf{root}_{\mathsf{Burn}}$
Entity S
Entity R

Figure 2: Process of the cross-chain transfer protocol Θ. The blue block depicts the hash digest of the transaction $\mathsf{Tx}_{\mathsf{Burn}}$ and the red block represents the Merkle proof information, i.e.,$\mathsf{h}_{\mathsf{Burn}}$ and $\mathsf{root}_{\mathsf{Burn}}$.

Assume that $\mathcal{S}^{\mathsf{I}}$ in Chain I sends amount $\mathsf{v}_{\mathcal{S}}$ to $\mathcal{R}^{\mathsf{II}}$ in Chain II, where $\mathsf{v}_{\mathcal{S}}$ is a standard value. The entire protocol consists of four steps: **Burn, Transmit, Mint, and Redeem**. Note that in a cross-chain transfer, the **Mint** and **Redeem** steps can only have one of them executed. If **Mint** is executed, it signifies a successful transfer; if **Redeem** is executed, it means the transferred amount has been returned to the sender.
Θ.Burn. In this step, $\mathcal{S}^{\mathsf{I}}$ sends the transaction $\mathsf{Tx}_{\mathsf{Burn}}$ to the smart contract ξ^{I} (or a dedicated burn-account) to burn $\mathsf{v}_{\mathcal{S}}$. The transaction $\mathsf{Tx}_{\mathsf{Burn}}$ contains the transfer amount $\mathsf{v}_{\mathcal{S}}$, and the hash digest $\mathsf{h}(\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}, \mathsf{r}, \mathsf{sn})$ calculated by hashing the receiver's public key $\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}$, and two random numbers r and sn. Using both r and sn in a hash function implementation can enhance the randomness of the hash result, hiding the receiver information $\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}$ and preventing brute force attacks [19]. Note that r and sn can be generated by utilizing a Pseudo-Random Function (PRF) [20]. Also, note that sn serves as a unique identifier (like the serial number in currency) for each cross-chain transfer activity. Once $\mathsf{v}_{\mathcal{S}}$ is claimed, the corresponding sn is publicly revealed to prevent double-spending attacks [42]. The smart contract ensures the legality of $\mathsf{Tx}_{\mathsf{Burn}}$, locks the asset $\mathsf{v}_{\mathcal{S}}$ after verifying the correctness of the sender's signature and whether or not the account balance is sufficient, and then sets a time lock T_3.

$$\mathsf{Tx}_{\mathsf{Burn}} \overset{\mathrm{def}}{=} (\mathsf{From}:\mathcal{S};\mathsf{To}:\xi;\mathsf{v}_{\mathcal{S}},\mathsf{h}(\widetilde{\mathsf{pk}}_{\mathcal{R}},\mathsf{r},\mathsf{sn})).$$

The above design ensures that an adversary can only get that a node sent the transaction $\mathsf{Tx}_{\mathsf{Burn}}$, without knowledge of its receiver.
Θ.Transmit. After reaching consensus, the hash digest of the transaction $\mathsf{Tx}_{\mathsf{Burn}}$ would be included as a leaf node in a block of Chain I. For simplicity, we omit non-essential data, such as the Nonce and block number, when describing the transaction hash. The hash digest of $\mathsf{Tx}_{\mathsf{Burn}}$ is defined as:

$$\mathsf{h}(\mathsf{Tx}_{\mathsf{Burn}}) \overset{\mathrm{def}}{=} \mathsf{hash}(\widetilde{\mathsf{pk}}_{\mathcal{S}},\mathsf{addr}_{\xi},\mathsf{v}_{\mathcal{S}},\mathsf{h}(\widetilde{\mathsf{pk}}_{\mathcal{R}},\mathsf{r},\mathsf{sn})).$$

Through off-chain communication, $\mathcal{S}$ sends to $\mathcal{R}$ the following parameters: random number r, random serial number sn, and the Merkle proof of $\mathsf{Tx}_{\mathsf{Burn}}$ consisting of a set of hashes, denoted by $\mathsf{h}_{\mathsf{Burn}}$, that includes the Merkle root $\mathsf{root}_{\mathsf{Burn}}$ and the Merkle path of $\mathsf{Tx}_{\mathsf{Burn}}$.

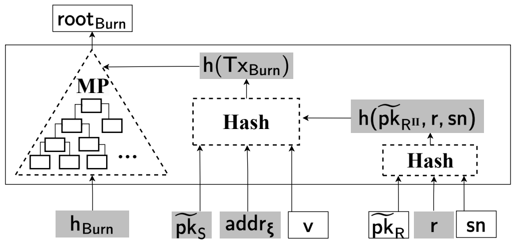


Figure 3: The circuit logic diagram for the privacy-preserving cross-chain transfer protocol. The parameters with a gray background are private ones protected with zk-SNARK.

Θ.Mint. After the off-chain communication, $\mathcal{R}^{\mathsf{II}}$ selects a security parameter and constructs a new circuit Λ_{Θ} (shown in Figure 3). The entire circuit includes two types of functions (represented by dashed boxes): Hash and MP, where Hash is used to calculate the hash value of the input data, while MP implements the Merkle proof functionality to prove that a transaction exists in a block. $\mathcal{R}^{\mathsf{II}}$ employs the algorithm Π.Setup (introduced in Section 4.2) to get the key pair $(\mathsf{pk}_{\Theta}, \mathsf{vk}_{\Theta})$. Then, it adopts the algorithm Π.Prove to generate a proof π_{Θ}, with the public inputs $(\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}, \mathsf{sn}, \mathsf{v}_{\mathcal{S}}, \mathsf{root}_{\mathsf{Burn}})$ and the private inputs $(\widetilde{\mathsf{pk}}_{\mathcal{S}^{\mathsf{I}}}, \mathsf{addr}_{\xi^{\mathsf{I}}}, \mathsf{r}, \mathsf{h}_{\mathsf{Burn}})$. Note that in this transaction, $\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}$ is public as the transaction is sent from $\mathcal{R}^{\mathsf{II}}$ to ξ^{II}. Next, $\mathcal{R}^{\mathsf{II}}$ packages π_{Θ} with the public inputs into the transaction $\mathsf{Tx}_{\mathsf{Mint}}$, and sends it to the smart contract ξ^{II} (or a dedicated mint-account) to acquire the assets of equal value to $\mathsf{v}_{\mathcal{S}}$. Finally, the smart contract ξ^{II} verifies the validity of π_{Θ} through algorithm Π.Verify and sends the asset to $\mathcal{R}^{\mathsf{II}}$.

$$\mathsf{Tx}_{\mathsf{Mint}} \overset{\text{def}}{=} (\mathsf{From}:\mathcal{R};\mathsf{To}:\xi;\pi,\widetilde{\mathsf{pk}}_{\mathcal{R}},\mathsf{sn},\mathsf{v},\mathsf{root}_{\mathsf{Burn}}).$$

Θ.Redeem. If $\mathcal{R}^{\mathsf{II}}$ fails to complete the step Θ.Mint before T_3, $\mathcal{S}^{\mathsf{I}}$ can request to redeem $\mathsf{v}_{\mathcal{S}}$ from the smart contract ξ^{I}. $\mathcal{S}^{\mathsf{I}}$ needs to create a proof as $\mathcal{R}^{\mathsf{II}}$ does in step Θ.Mint, with the only difference being that $\widetilde{\mathsf{pk}}_{\mathcal{S}^{\mathsf{I}}}$ and $\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}$ must be changed to public and private inputs, respectively, in the circuit design. Upon completion of the proof, $\mathcal{S}^{\mathsf{I}}$ sends $\mathsf{Tx}_{\mathsf{Redeem}}$ to ξ^{I} and receives $\mathsf{v}_{\mathcal{S}}$ after a successful validation.

$$\mathsf{Tx}_{\mathsf{Redeem}} \overset{\text{def}}{=} (\mathsf{From}:\mathcal{S};\mathsf{To}:\xi;\pi,\widetilde{\mathsf{pk}}_{\mathcal{S}},\mathsf{sn},\mathsf{v},\mathsf{root}_{\mathsf{Burn}}).$$

5.2.2 An Cross-chain Exchange Protocol with Privacy Preservation

We employ HTLC to develop an exchange protocol. According to the definition of HTLC in Section 4.3, it is evident that because HTLC.Unlock requires both parties in cross-chain interactions to use the same preimage to unlock the exchanged assets, attackers can easily identify the correlation between the two parties. Therefore our design challenge for privacy protection is to obscure the preimage while ensuring the atomicity of the HTLC protocol. Note that we also need to protect the privacy of the exchanged values. For this purpose, we take the same idea as that for the cross-chain transfer by setting the fundamental denominations to eliminate the risk of exposing privacy through exchanged amounts.

To address the privacy challenges posed by the preimage in HTLC, we develop a privacy-preserving protocol Φ based on zk-SNARK (introduced in Section 4.2). The protocol consists of four steps: **Prepare, Lock, Unlock, and Refund**. According to the requirement of HTLC (shown in Section 4.3), Φ necessitates the existence of accounts for $\mathcal{S}$ and $\mathcal{R}$ on both Chain I (i.e., $\mathcal{S}^{\mathsf{I}}$, $\mathcal{R}^{\mathsf{I}}$) and Chain II (i.e., $\mathcal{S}^{\mathsf{II}}$, $\mathcal{R}^{\mathsf{II}}$). Note that in a cross-chain exchange, only one of the **Unlock** and **Refund** steps can be executed. The execution of **Unlock** indicates a successful exchange, while the execution of **Refund** implies that the exchanged amount has been returned to the sender.

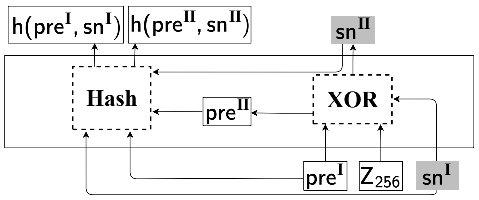


Figure 4: The circuit logic diagram used for Φ.Prepare. The parameters with a gray background are private ones protected with zk-SNARK.

Φ.Prepare. During this process, $\mathcal{S}$ is required to generate a security parameter and then employs it, along with a circuit $\Lambda_{\Phi}^{\mathsf{off}}$ illustrated in Figure 4, to derive the key pair $(\mathsf{pk}_{\Phi}^{\mathsf{off}}, \mathsf{vk}_{\Phi}^{\mathsf{off}})$ based on the setup algorithm Π.Setup. The circuit $\Lambda_{\Phi}^{\mathsf{off}}$ comprises of two functions (indicated by dashed lines), namely a hash function denoted as Hash and an exclusive OR operation denoted as XOR. Specifically, $\mathcal{S}$ generates two preimages ($\mathsf{pre}^{\mathsf{I}}$ and $\mathsf{pre}^{\mathsf{II}}$), two random serial number (sn^{I} and $\mathsf{sn}^{\mathsf{II}}$), and a 256-bit integer Z_{256}, where sn^{I} and Z_{256} are combined with XOR to derive $\mathsf{sn}^{\mathsf{II}}$. These random serial numbers are kept as private inputs, while preimages and Z_{256} are used as public inputs. Making use of the Hash function, $\mathsf{h}(\mathsf{pre}^{\mathsf{I}}, \mathsf{sn}^{\mathsf{I}})$ and $\mathsf{h}(\mathsf{pre}^{\mathsf{II}}, \mathsf{sn}^{\mathsf{II}})$ are calculated by taking ($\mathsf{pre}^{\mathsf{I}}$, sn^{I}), and ($\mathsf{pre}^{\mathsf{II}}$, $\mathsf{sn}^{\mathsf{II}}$) as inputs, respectively. Then, by employing the protocol Π.Prove, $\mathcal{S}$ generates a proof $\pi_{\Phi}^{\mathsf{off}}$. The proof along with the public inputs, such as the hash results ($\mathsf{h}(\mathsf{pre}^{\mathsf{I}}, \mathsf{sn}^{\mathsf{I}})$, $\mathsf{h}(\mathsf{pre}^{\mathsf{II}}, \mathsf{sn}^{\mathsf{II}})$), preimages ($\mathsf{pre}^{\mathsf{I}}$, $\mathsf{pre}^{\mathsf{II}}$), and the integer Z_{256}, are sent to $\mathcal{R}$ by $\mathcal{S}$ through off-chain communications. This off-chain process can achieve the following goals: first, it allows $\mathcal{S}$ to prove the relationship between sn^{I} and $\mathsf{sn}^{\mathsf{II}}$ without disclosing sn^{I}; second, it enables $\mathcal{R}$ to easily derive sn^{I} from $\mathsf{sn}^{\mathsf{II}}$ through a reverse operation.

Φ.Lock. After off-chain communications, based on $\mathsf{Tx}_{\mathsf{Lock}}$,

$\mathcal{S}^{\mathsf{I}}$ uses $\mathsf{h}(\mathsf{pre}^{\mathsf{I}}, \mathsf{sn}^{\mathsf{I}})$ to lock the asset $\mathsf{v}_{\mathcal{S}}$ in the smart contract ξ^{I}. After $\mathcal{R}^{\mathsf{I}}$ verifies the correctness of $\mathsf{Tx}_{\mathsf{Lock}}$ sent by $\mathcal{S}^{\mathsf{I}}$, $\mathcal{R}^{\mathsf{II}}$ sends $\mathsf{Tx}_{\mathsf{Lock}}$ with $\mathsf{h}(\mathsf{pre}^{\mathsf{II}}, \mathsf{sn}^{\mathsf{II}})$ to lock the amount $\mathsf{v}_{\mathcal{R}}$ in the smart contract ξ^{II}. The time lock operation involved is consistent with the step HTLC.Lock (introduced in Section 4.3), where both parties set time locks T_1 and T_2 on smart contracts, with $\mathsf{T}_1 > \mathsf{T}_2$.

$$\mathsf{Tx}_{\mathsf{Lock}} \overset{\text{def}}{=} (\mathsf{From}: \mathcal{S}/\mathcal{R}; \mathsf{To}: \xi; \mathsf{v}, \mathsf{h}(\mathsf{pre}, \mathsf{sn}, \mathsf{v})).$$

After reaching consensus, $\mathsf{Tx}_{\mathsf{Lock}}$ would be included as a leaf node in a block and the hash digest of $\mathsf{Tx}_{\mathsf{Lock}}$ is defined as:

$$\mathsf{h}(\mathsf{Tx}_{\mathsf{Lock}}) \overset{\text{def}}{=} \mathsf{hash}(\widetilde{\mathsf{pk}}, \mathsf{addr}_{\xi}, \mathsf{v}, \mathsf{h}(\mathsf{pre}, \mathsf{sn})).$$

Φ.Unlock. Within T_2, $\mathcal{S}^{\mathsf{II}}$ first generates a circuit $\Lambda_{\Phi}^{\mathsf{on}}$ that has similar logic to Λ_{Θ} (shown in Figure 3) but requires modifying its inputs (shown in Figure 5). Specifically, the public inputs include the serial number $\mathsf{sn}^{\mathsf{II}}$, the amount $\mathsf{v}_{\mathcal{R}}$, and the Merkle root $\mathsf{root}^{\mathsf{II}}_{\mathsf{Lock}}$. The private inputs consist of the $\mathcal{R}^{\mathsf{II}}$'s public key $\widetilde{\mathsf{pk}}_{\mathcal{R}^{\mathsf{II}}}$, the address of the smart contract $\mathsf{addr}_{\xi^{\mathsf{II}}}$, the preimage $\mathsf{pre}^{\mathsf{II}}$, the Merkle proof of $\mathsf{Tx}_{\mathsf{Lock}}$ (denoted by $\mathsf{h}^{\mathsf{II}}_{\mathsf{Lock}}$), the transaction hash $\mathsf{h}^{\mathsf{II}}(\mathsf{Tx}_{\mathsf{Lock}})$, and the hash lock $\mathsf{h}(\mathsf{pre}^{\mathsf{II}}, \mathsf{sn}^{\mathsf{II}})$. Similar to Θ.Mint, $\mathcal{S}^{\mathsf{II}}$ generates a proof π^{II}_{Φ} based on the above circuit. Then, $\mathcal{S}^{\mathsf{II}}$ sends π^{II}_{Φ} along with the public inputs to the smart contract ξ^{II} through $\mathsf{Tx}_{\mathsf{Unlock}}$ to unlock $\mathsf{v}_{\mathcal{R}}$. After that, $\mathcal{R}^{\mathsf{II}}$ learns $\mathsf{sn}^{\mathsf{II}}$ and uses the XOR operation with Z_{256} to obtain sn^{I}. Following the same steps as $\mathcal{S}^{\mathsf{II}}$, $\mathcal{R}^{\mathsf{I}}$ generates a proof π^{I}_{Φ} with public inputs (sn^{I}, $\mathsf{v}_{\mathcal{S}}$, $\mathsf{root}^{\mathsf{I}}_{\mathsf{Lock}}$) and private inputs ($\widetilde{\mathsf{pk}}_{\mathcal{S}^{\mathsf{I}}}$, $\mathsf{addr}_{\xi^{\mathsf{I}}}$, $\mathsf{pre}^{\mathsf{I}}$, $\mathsf{h}^{\mathsf{I}}_{\mathsf{Lock}}$, $\mathsf{h}^{\mathsf{I}}(\mathsf{Tx}_{\mathsf{Lock}})$, $\mathsf{h}(\mathsf{pre}^{\mathsf{I}}, \mathsf{sn}^{\mathsf{I}})$), then sends $\mathsf{Tx}_{\mathsf{Unlock}}$ to get $\mathsf{v}_{\mathcal{S}}$.

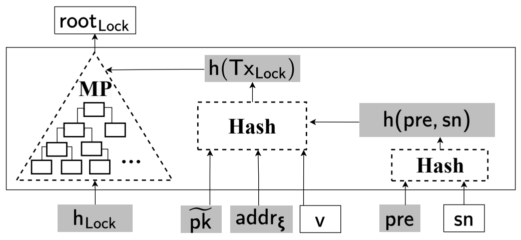
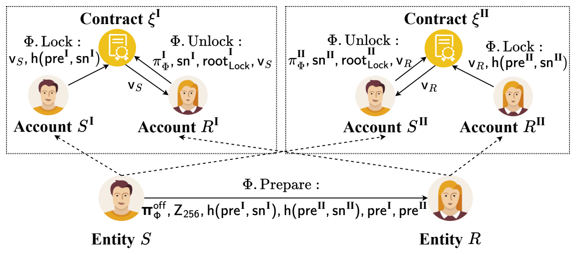


Figure 5: The circuit logic diagram used for Φ.Unlock. The parameters with a gray background are private ones protected with zk-SNARK.

$$\mathsf{Tx}_{\mathsf{Unlock}} \overset{\text{def}}{=} (\mathsf{From}: \mathcal{S}/\mathcal{R}; \mathsf{To}: \xi; \pi, \mathsf{sn}, \mathsf{v}, \mathsf{root}_{\mathsf{Lock}}).$$

Φ.Refund. As in HTLC.Refund (introduced in Section 4.3), if $\mathcal{S}^{\mathsf{II}}$ fails to send $\mathsf{Tx}_{\mathsf{Unlock}}$ within T_2, $\mathcal{R}^{\mathsf{II}}$ would be unable to calculate sn^{I}. Therefore, the smart contracts on both chains would return the locked assets after the timeout.

For the convenience of the readers' understanding, we summarize the process of the exchange between $\mathcal{S}$ and $\mathcal{R}$ using protocol Φ in Figure 6.



Contract $\xi^{\mathbf{I}}$
Φ. Lock : v_S, $\mathsf{h}(\mathsf{pre}^{\mathbf{I}}, \mathsf{sn}^{\mathbf{I}})$
Φ. Unlock : $\pi^{\mathbf{I}}_{\Phi}$, $\mathsf{sn}^{\mathbf{I}}$, $\mathsf{root}^{\mathbf{I}}_{\mathsf{Lock}}$, v_S
v_S
Account $S^{\mathbf{I}}$
Account $R^{\mathbf{I}}$
Contract $\xi^{\mathbf{II}}$
Φ. Unlock : $\pi^{\mathbf{II}}_{\Phi}$, $\mathsf{sn}^{\mathbf{II}}$, $\mathsf{root}^{\mathbf{II}}_{\mathsf{Lock}}$, v_R
Φ. Lock : v_R, $\mathsf{h}(\mathsf{pre}^{\mathbf{II}}, \mathsf{sn}^{\mathbf{II}})$
v_R
Account $S^{\mathbf{II}}$
Account $R^{\mathbf{II}}$
Φ. Prepare :
$\pi^{\mathsf{off}}_{\Phi}$, Z_{256}, $\mathsf{h}(\mathsf{pre}^{\mathbf{I}}, \mathsf{sn}^{\mathbf{I}})$, $\mathsf{h}(\mathsf{pre}^{\mathbf{II}}, \mathsf{sn}^{\mathbf{II}})$, $\mathsf{pre}^{\mathbf{I}}$, $\mathsf{pre}^{\mathbf{II}}$
Entity S
Entity R

Figure 6: Process of the cross-chain exchange based on protocol Φ.

5.2.3 Discussion

When contingencies are in place for parties not following the protocols (Θ or Φ) explicitly, the protocol also guarantees the correctness of the amount settlement.

The protocol Θ ensures that the interaction results in either $\mathcal{R}$ receiving $\mathsf{v}_{\mathcal{S}}$ or $\mathcal{S}$ revoking $\mathsf{v}_{\mathcal{S}}$. Specifically, after $\mathcal{S}$ completes Θ.Burn, if $\mathcal{S}$ fails to execute Θ.Transmit, $\mathcal{R}$ cannot perform Θ.Mint. To prevent the deadlock of the amount $\mathsf{v}_{\mathcal{S}}$, $\mathcal{S}$ can execute Θ.Redeem to reclaim $\mathsf{v}_{\mathcal{S}}$ after a timeout if $\mathcal{R}$ does not execute Θ.Mint.

The protocol Φ ensures that the interaction results in both $\mathcal{S}$ and $\mathcal{R}$ receiving each other's funds or both parties refunding their respective funds. In Φ.Lock, if party $\mathcal{S}$ executes the lock operation but party $\mathcal{R}$ does not, $\mathcal{S}$ would reject executing Φ.Unlock and $\mathsf{sn}^{\mathsf{II}}$ remain undisclosed. In this case, $\mathcal{R}$ cannot calculate sn^{I} and thus cannot execute Φ.Unlock. Consequently, during Φ.Refund, both parties would refund their respective funds. Once $\mathcal{S}$ executes Φ.Unlock, $\mathcal{R}$ can receive $\mathsf{sn}^{\mathsf{II}}$ and calculate sn^{I} to complete the unlock operation. In this case, both parties can obtain the amount of the other party.

5.3 Cross-chain Auditing Protocol

To solve the FAI issue, we develop an auditing protocol Ψ, which enhances the efficiency of cross-chain auditing. Moreover, Ψ can be compatible with privacy protection protocols (introduced in Section 5.2), thereby addressing the IPA problem.

In our scheme, the purpose of auditing is to check whether transactions within blocks of the lower-layer blockchain meet the auditing requirements established by the auditors. Depending on specific requirements, auditing tasks can be categorized into basic tasks or complex tasks. The former involves auditing individual transactions, while the latter involves auditing multiple transactions collectively. zkCross can be used for both types of auditing tasks: basic tasks, such as verifying the legitimacy of transaction amounts and addresses, and complex tasks, such as confirming the legitimacy of total amounts sent by each node over a specified time period. The core of the protocol Ψ lies in a newly proposed circuit (shown in Figure 8) that simultaneously considers transaction auditing and aggregation. The aggregation process is implemented in an approach that is similar to zk-Rollup (introduced in Section 4.2), except that our solution can further reduce the on-chain storage cost and hide the transaction content. The original intention of zk-Rollup is to handle off-chain transactions that have not undergone consensus. In zkCross, the focus of the audit is on transactions that have been validated by consensus in the lower-layer blockchains. As a result, in certain auditing tasks, there is no need to disclose transaction amounts and the associated addresses, as done in zk-Rollup, to prevent malicious tampering of the off-chain transactions.

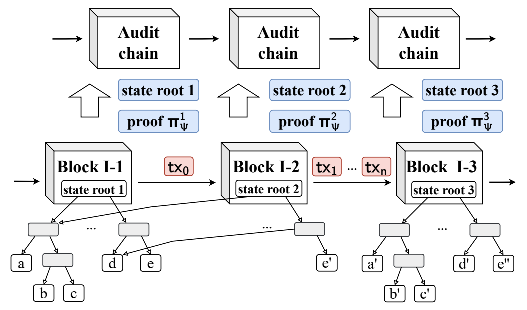


Figure 7: An example interaction between the audit chain and Chain I based on the cross-chain auditing protocol.

To aid in comprehension, we expound on the details of protocol Ψ through the example depicted in Figure 7. This example displays three blocks (identified as Block I-1–Block I-3) of Chain I, and the states of Chain I are denoted as a–e in the first block. Whenever miners create a new block, such as Block I-2 that contains a transaction tx_0, which modifies the account state from e to e′, the corresponding state root changes from root 1 to root 2. The majority of blockchain systems involve many transactions and alterations in the states of multiple accounts, such as the transition from Block I-2 to Block I-3. Upon discovering a new block, the committer updates the new state root of Chain I, i.e., state root 3, on the audit chain. Simultaneously, to demonstrate the correctness of the state transition to the auditors, the committer generates a zero-knowledge proof, such as the proof π_Ψ^3 shown in Figure 7. The entire process of protocol Ψ can be divided into three steps: **Initialize, Commit, and Audit**.

Ψ.Initialize. To initiate the protocol Ψ based on algorithm Π.Setup, the committer $\mathcal{C_T}$ is required to generate a security parameter, denoted as 1^λ, and subsequently utilizes 1^λ along with a circuit Λ_Ψ illustrated in Figure 8 to derive the key pair $(\mathsf{pk}_\Psi, \mathsf{vk}_\Psi)$. One can observe that circuit Λ_Ψ consists of four functions (indicated by dashed lines), i.e., the Auditing Function AF, the Signature Verification Function SVF, the State Transition Function STF, and the Root Verification Function RVF. In this example, the auditing focus is on validating the legitimacy of the sender's address within Chain I. Therefore, AF audits transactions by checking if the address is present in the blacklist, thus completing transaction auditing. SVF is designed to demonstrate the correctness of the transaction signatures; STF ensures the correctness of the transition from the old state $\mathsf{State}^{\mathsf{old}}$ to the new state $\mathsf{State}^{\mathsf{new}}$ after the transaction takes place; and RVF guarantees the consistency of $\mathsf{State}^{\mathsf{old}}$ and $\mathsf{State}^{\mathsf{new}}$ with the states recorded in the blocks of Chain I by recomputing the state root. For example, in Figure 7, the state of Block I-2 ($\mathsf{State}^{\mathsf{old}} = \mathsf{a,b,c,d,e'}$) transitions to the state of Block I-3 ($\mathsf{State}^{\mathsf{new}} = \mathsf{a',b',c',d',e''}$) after n transactions. Note that, one can compress transactions from multiple blocks and modify the logic of the circuit Λ_Ψ according to the auditing task.

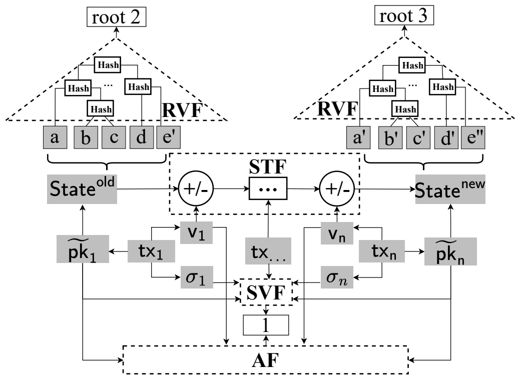


Figure 8: The logic diagram of the circuit used for the cross-chain auditing protocol Ψ. The specific values of the parameters are derived from the state transition process between Block I-2 to Block I-3, as illustrated in Figure 7. Private inputs are indicated by a gray background.

Ψ.Commit. In this step, the committer $\mathcal{C_T}$ requires information to set specific private and public inputs for the algorithm Π.Prove. We continue to use Block I-3 as an example, as depicted in Figure 7. Specifically, $\mathcal{C_T}$ gathers a set of transactions (tx_1–tx_n) and sets the account state information ($\mathsf{State}^{\mathsf{old}}, \mathsf{State}^{\mathsf{new}}$) with transaction content (such as transfer amount v, public key $\widetilde{\mathsf{pk}}$, and signature σ) as the private input $\vec{\mathsf{w}}$, while the public input $\vec{\mathsf{x}}$ includes the initial root (root 2), and the final state root (root 3). Note that, the blacklist can be directly written into the circuit as a constant, thereby reducing the number of circuit inputs. Subsequently, $\mathcal{C_T}$ invokes Π.Prove based on the aforementioned data $(\vec{\mathsf{x}};\vec{\mathsf{w}})$ and the proving key pk generated by Ψ.Initialize to produce the proof π_Ψ^3 for the corresponding state transition. Finally, $\mathcal{C_T}$ registers as a light node in the audit chain and packages public inputs $\vec{\mathsf{x}}$ with π_Ψ^3 into a new transaction $\mathsf{Tx}_{\mathsf{Commit}}$ to the audit chain.

$$\mathsf{Tx}_{\mathsf{Commit}} \overset{\text{def}}{=} (\mathsf{From} : \mathcal{C_T}; \mathsf{To} : \xi; \vec{\mathsf{x}}, \pi).$$

Ψ.Audit. After receiving the transaction $\mathsf{Tx}_{\mathsf{Commit}}$, auditors would call Π.Verify based on parameters vk, $\vec{\mathsf{x}}$, and π_Ψ^3 to verify and reach a consensus on the proof. Upon a successful consensus, the latest uploaded state root (root 3) would be stored as a leaf in the state tree of the audit chain. Of course, this step can also be written into the smart contract of the audit chain to reduce the security risks caused by human errors. Note that, auditors can collect multiple proofs for verification and auditing, enabling them to conduct certain complex auditing tasks, such as auditing the total amount of transactions sent by a node across multiple blocks.

6 Security Analysis

In this section, we analyze that zkCross can achieve the goals of privacy and efficiency (as outlined in Section 4.1).

Lemma 1 (Properties of zk-SNARK [13,15]). *zk-SNARK has the property of succinctness, where the complexity of verifying a proof remains constant at $O(1)$ regardless of the complexity of the proving process. Moreover, the proof size remains fixed. Additionally, zk-SNARK is a non-interactive zero-knowledge proof technique that satisfies completeness, soundness, and zero-knowledge requirements.*

Theorem 1 (Unlinkability). *A sender $\mathcal{S}$ interacts with a receiver $\mathcal{R}$ via protocol Θ or Φ. Given the assurance of security offered by the collision-resistant hash function and the zero-knowledge proof system, within an anonymity set, the adversary cannot link $\mathcal{R}$ based on the information initiated by $\mathcal{S}$, nor vice versa.*

Proof. zkCross considers all factors that may compromise the unlinkability, including receiver addresses, preimages, and transaction amounts. Specifically, each transaction in blockchains records the addresses of both sender and receiver, providing adversaries with information to link both parties. Additionally, in HTLCs, both parties use the same preimage to generate hash locks, facilitating adversaries in linking their accounts. Moreover, the variation in the amount of each transaction allows adversaries to link the accounts of both sender and receiver with the same state changes. Within an anonymity set, we use the following experiment to define the unlinkability: Given two transactions sent by $\mathcal{S}$, one (denoted as tx) to $\mathcal{R}$, and the other (denoted as tx′) to $\mathcal{R}'$, the adversary $\mathcal{A}$ is guessing whether the transaction tx or tx′ is sent for $\mathcal{R}$. Note that tx and tx′ are two transactions of the same type, for example, both are $\mathsf{Tx}_{\mathsf{Burn}}$. The probability of $\mathcal{A}$'s success is defined as $\Pr[\mathbf{Exp}^{\mathsf{Unlinkability}}_{\mathcal{A},\mathsf{zkCross}}(\lambda)=1]$, where λ is the security parameter. zkCross satisfies the unlinkability if, for all adversaries, it holds that $|\Pr[\mathbf{Exp}^{\mathsf{Unlinkability}}_{\mathcal{A},\mathsf{zkCross}}(\lambda)=1]-\frac{1}{2}|\leq \mathsf{negl}(\lambda)$.

We denote $\mathbb{U}$ as an anonymity set consisting of the receivers whose states change during a period, among whom the actual receiver of a transaction is concealed. The size of an anonymity set is denoted as $|\mathbb{U}|$. If adversaries know a sender's address and intend to compromise unlinkability, they must link it to the corresponding receiver's address from an anonymity set $\mathbb{U}$. If $|\mathbb{U}|$ is 1, there is only one active pair of $\mathcal{S}$ and $\mathcal{R}$. Therefore, concealing the relation between $\mathcal{S}$ and $\mathcal{R}$ is impossible in our context through the utilization of any privacy-preserving solutions [8,12,26,27,29]. Therefore, in the following context, we only consider an anonymity set $\mathbb{U}$ with $|\mathbb{U}|>1$, zkCross employs methods such as hiding receiver addresses, utilizing independent preimages, and adding transactions with same amounts, thereby guaranteeing that the probability of adversaries compromising unlinkability is negligible.

[Hide receiver address] zkCross employs hashing and zk-SNARK to hide the receiver's address.

Specifically, in Θ.Burn, zkCross employs $\mathsf{h}(\widetilde{\mathsf{pk}_{\mathcal{R}}},\mathsf{r},\mathsf{sn})$ to hide the receiver's address $\widetilde{\mathsf{pk}_{\mathcal{R}}}$. To obtain $\widetilde{\mathsf{pk}_{\mathcal{R}}}$, an adversary must get sn and r, where sn is revealed at the end of the protocol, and r is a private input protected by the zk-SNARK. According to Lemma 1, zk-SNARK has the property of zero-knowledge, meaning that the probability of $\mathcal{A}$ can derive the private input r is negligible. Moreover, due to the security of the hash function, the probability that $\mathcal{A}$ obtains $\widetilde{\mathsf{pk}_{\mathcal{R}}}$ from $\mathsf{h}(\widetilde{\mathsf{pk}_{\mathcal{R}}},\mathsf{r},\mathsf{sn})$ without knowing r is negligible. During Θ.Mint, $\mathcal{R}^{\mathsf{II}}$ takes $\widetilde{\mathsf{pk}_{\mathcal{S}}}$ as the private input for the zk-SNARK to hide $\mathcal{S}^{\mathsf{I}}$'s address. Similarly, in Θ.Redeem, $\mathcal{S}^{\mathsf{I}}$ takes $\widetilde{\mathsf{pk}_{\mathcal{R}}}$ as the private input to hide $\mathcal{R}^{\mathsf{II}}$'s address, which indicates that $\mathcal{A}$ should break the zero-knowledge property of the zk-SNARK and this probability is negligible.

During cross-chain exchanges, the protocol Φ adopts HTLC, which does not involve interaction between $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{II}}$. HTLC involves only the transfer of funds between accounts within the same chain, such as between $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{I}}$, or between $\mathcal{S}^{\mathsf{II}}$ and $\mathcal{R}^{\mathsf{II}}$. Φ first adopts a similar approach to the protocol Θ in hiding the receiver's address, preventing adversaries from linking the addresses of both interacting parties within the same chain (such as $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{I}}$). Moreover, since the accounts of the same entity on different chains are independent, adversaries cannot link between $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{S}^{\mathsf{II}}$, or between $\mathcal{R}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{II}}$. Similar to the analysis for protocol Θ, based on the security of zk-SNARKs and hash functions, the probability of $\mathcal{A}$ deducing the receiver address is negligible. Based on the above design, in $\mathbf{Exp}^{\mathsf{Unlinkability}}_{\mathcal{A},\mathsf{zkCross}}(\lambda)=1$, the probability of the adversary $\mathcal{A}$ linking $\mathcal{S}$ and $\mathcal{R}$ based on the receiver address is $\frac{1}{2}+\mathsf{negl}(\lambda)$.

[Use independent preimages] In HTLCs, when $\mathcal{S}$ interacts with $\mathcal{R}$, this interaction involves two same hash locks generated by a preimage. $\mathcal{A}$ can link $\mathcal{S}$ and $\mathcal{R}$ based on the same hash locks. The interacting parties in zkCross use independent preimages to generate hash locks and adopt zk-SNARK to ensure the correctness of preimages.

Specifically, in step Φ.Lock, $\mathcal{S}^{\mathsf{I}}$ and $\mathcal{R}^{\mathsf{II}}$ use independent hash locks to lock the exchange amounts. However, using independent preimages may compromise the security of the protocol, for example, $\mathcal{R}$ cannot verify whether the preimage provided by $\mathcal{S}$ is correct. The design challenge lies in the entity $\mathcal{S}$ needing to convince $\mathcal{R}$ of the correctness of the hash lock and ensuring that $\mathcal{R}^{\mathsf{I}}$ executes the unlock operation only after $\mathcal{S}^{\mathsf{II}}$ has executed the unlock operation. To solve the above challenge, we design a novel circuit shown in Figure 4. Based on this circuit, $\mathcal{S}$ can generate a proof to convince $\mathcal{R}$ of the correctness of the hash lock, thereby eliminating the assumption that both parties need to share the same secret [6]. Since the unlocking information sn is hidden, $\mathcal{R}^{\mathsf{I}}$ can only execute the unlock operation after $\mathcal{S}^{\mathsf{II}}$ has executed the unlock operation (reveal sn). Based on the above design, given two hash locks generated by two independent preimages, the probability that the adversary breaks the collision resistance of the hash function to link two hash locks is negligible. Therefore, in $\mathbf{Exp}^{\mathsf{Unlinkability}}_{\mathcal{A},\mathsf{zkCross}}(\lambda) = 1$, the probability that the adversary links $\mathcal{S}$ and $\mathcal{R}$ based on preimages is $\frac{1}{2} + \mathsf{negl}(\lambda)$.

[Add transactions with same amounts] We establish fundamental denominations to divide a transfer amount into multiple sub-amounts, thereby increasing the number of transactions with the same amounts within the network. Similar to works [12, 16, 29], the sub-amount is a fixed value that can be set according to specific scenarios. The above design means that one can always find two transactions with the same amount from the network, ensuring that $|\mathbb{U}| > 1$.

Based on the above analysis, one can conclude that in the experiment $\mathbf{Exp}^{\mathsf{Unlinkability}}_{\mathcal{A},\mathsf{zkCross}}(\lambda) = 1$, the probability that an adversary compromises unlinkability, even when collecting all factors that may compromise unlinkability such as receiver addresses, preimages, and transaction amounts, is $\frac{1}{2} + \mathsf{negl}(\lambda)$. Therefore, $|\Pr[\mathbf{Exp}^{\mathsf{Unlinkability}}_{\mathcal{A},\mathsf{zkCross}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$, indicating that $\mathsf{zkCross}$ satisfies unlinkability.

□

Theorem 2 (Efficiency)**.** *In a network consisting of* k *ordinary blockchains, each blockchain has an average block production rate of* m *new blocks per second and processes an average of* n *transactions per block.* $\mathsf{zkCross}$ *can increase the audit efficiency by nearly* n *times.*

Proof. We consider two auditing methods: auditors examining each transaction individually (referred to as the full auditing method) with an average audit time of t_1 per transaction, or relying on protocol Ψ for auditing with an average verification time of t_2 per proof. Based on Lemma 1, zk-SNARK exhibits succinctness, ensuring that t_1 and t_2 remain constant at $O(1)$. The first approach would require $\mathsf{t}_1 \times \mathsf{k} \times \mathsf{m} \times \mathsf{n}$ to audit the entire network, while the second one would demand $\mathsf{t}_2 \times \mathsf{k} \times \mathsf{m}$. The ratio between the two methods is $\frac{\mathsf{t}_1 \times \mathsf{n}}{\mathsf{t}_2}$. Notably, as the number of transactions processed within a block increases linearly, the time growth of verifying proofs in the Ψ.Audit step remains constant. Consequently, $\mathsf{zkCross}$ is nearly n times faster in audit time compared to the full auditing scheme for the same task. □

7 Performance Evaluation

In this section, we implement zkCross systems with both local and cloud servers and conduct comprehensive tests to evaluate the performance of the three protocols in terms of run time, latency, throughput, gas consumption, audit time, and proof size. Our code repository is open-sourced on Github[1].

7.1 Implementation

zkCross has three protocols, i.e., the two privacy-preserving cross-chain protocols (Θ, Φ) and the cross-chain auditing protocol Ψ, with each including two critical components: the off-chain zk-SNARK and the on-chain transaction. For the off-chain zk-SNARK, we use xjsnark[2] to obtain the circuits in each protocol, which are implemented with 1,500+ lines of code. Then, we utilize the Groth16 algorithm [13] to realize the Setup (initialization), Prove (generation), and Verify (verification) of zk-SNARK. The Groth16 algorithm is well-known and has been established as a reliable paradigm in blockchain [14]. For the on-chain component, each node runs go-ethereum[3] (an official implementation of the Ethereum protocol using Golang) to build the blockchain, and we employ Solidity[4] to write smart contracts and implement the transactions with 500+ lines of Solidity. To address the limitation of go-ethereum and Solidity of not supporting the verification of zk-SNARK, we make modifications to the source code of go-ethereum and Solidity, adding a function to smart contracts for proof verification, and setting the gas required to invoke this function at 440,000 gas (equivalent to Groth16 [36]).

We test the performance of zkCross on both local and cloud servers. The local servers are equipped with an Intel® Xeon(R) Silver 4214R CPU @ 2.40 GHz * 48 and 98.3 GB RAM running 64-bit Ubuntu 20.04.2 LTS. As for the experiment on cloud servers, we use 50 ecs.g6.3xlarge instances, with each running the Ubuntu 20.04 system Intel Xeon (Cascade Lake) Platinum 8269CY processor and having 12 vCPUs of frequency 3.2 GHz and 48 GB RAM. We start 4 docker nodes in each instance to form multiple blockchains based on the Proof-of-Work consensus algorithm, and the number of transactions at each blockchain reaches up to 10,000. More details of the setup of each experiment will be presented before reporting the performance evaluation results.

7.2 Experimental Results

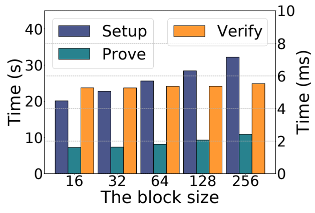
7.2.1 Cross-Chain Transfer and Exchange

We build two blockchains on cloud servers, varying the number of nodes and transactions in each blockchain based on the

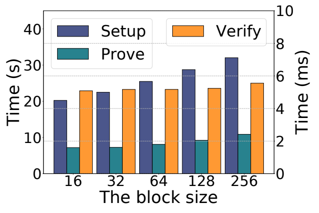
[1]https://github.com/Anonymous-Authors-zkCross/zkCross
[2]https://github.com/akosba/xjsnark
[3]https://github.com/ethereum/go-ethereum
[4]https://github.com/ethereum/solidity

test requirements.

We first test the performance of zk-SNARK, including the time consumption in three steps: Setup, Prove, and Verify. The performance of zk-SNARK varies based on the circuit used. In protocols Θ and Φ, there are a total of three circuits. Circuit Λ_Θ (as depicted in Figure 3) is used for protocol Θ, while circuits $\Lambda_\Phi^{\mathsf{off}}$ (as depicted in Figure 4) and $\Lambda_\Phi^{\mathsf{on}}$ (as depicted in Figure 5) are used for protocol Φ. The size of Λ_Θ and $\Lambda_\Phi^{\mathsf{on}}$ is related to the inputs of the MP function, which corresponds to the number of transactions included within a given block, i.e., the block size. For instance, a block containing 16 transactions (block size = 16) can construct a full binary tree with a height of 4, requiring MP to consist of 4 hash values with corresponding 4 hash operations.



(a)



(b)

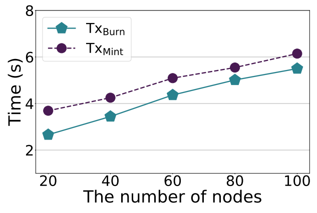
Figure 9: Run time for the initialization (Setup), generation (Prove), and verification (Verify) of proofs with different block sizes. Left: Λ_Θ; right: $\Lambda_\Phi^{\mathsf{on}}$.

In Figure 9, we test the time consumption of zk-SNARK related to circuits Λ_Θ and $\Lambda_\Phi^{\mathsf{on}}$, and change the block size from 16 to 256. As the block size increases, the time consumption of Setup and Prove increases linearly, while Verify remains nearly unchanged. While Setup and Prove take more time when compared to the millisecond-level Verify, they do not require additional on-chain resources as they run off-chain. Table 1 shows the average time consumption of Setup, Prove, and Verify related to $\Lambda_\Phi^{\mathsf{off}}$, with values of 6.96 seconds, 1.91 seconds, and 5.16 milliseconds, respectively.

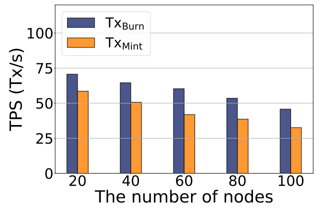
Table 1: Run time of the three steps based on $\Lambda_\Phi^{\mathsf{off}}$.

Setup (s)	Prove (s)	Verify (ms)
6.96	1.91	5.16

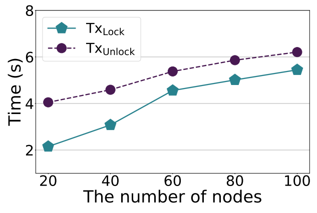
Then, we test the latency of transactions and system throughput at different node scales. The protocol Θ encompasses transactions $\mathsf{Tx}_{\mathsf{Burn}}$ and $\mathsf{Tx}_{\mathsf{Mint}}$, and $\mathsf{Tx}_{\mathsf{Redeem}}$ ($\mathsf{Tx}_{\mathsf{Redeem}}$ exhibiting the same performance as $\mathsf{Tx}_{\mathsf{Mint}}$), while the protocol Φ integrates transactions $\mathsf{Tx}_{\mathsf{Lock}}$ and $\mathsf{Tx}_{\mathsf{Unlock}}$. Note that the transaction delay refers to the duration between the time a transaction is sent to the blockchain and the time it is confirmed by miners. The unit utilized to measure the throughput is TPS, which refers to the number of transactions a scheme can process per second. For more accurate tests, we establish two blockchains and vary the number of on-chain nodes to simulate cross-chain environments. As shown in Figures 10, the maximum number of nodes for each blockchain is 100. The number of transactions in the network is related to the number of nodes, that is, the more nodes, the more transactions in the network. We stipulate that the number of transactions is equal to 10 times that of the nodes, i.e., there are 1,000 pending transactions in a network of 100 nodes.



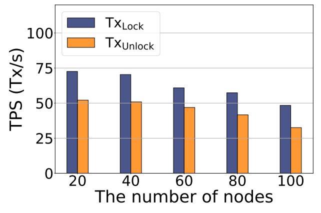
(a)



(b)



(c)



(d)

Figure 10: Network latency (left) and throughput (right) of the cross-chain transfer protocol Θ (top) and exchange protocol Φ (bottom).

Specifically, Figures 10 ((a), (b)) and ((c), (d)) respectively show the performance of protocols Θ and Φ. As the number of nodes in the network increases, the latency of the four types of transactions slightly increases, leading to a slight decrease in the corresponding network throughput. When there are 100 nodes in the network, the delay of the transaction $\mathsf{Tx}_{\mathsf{Burn}}$ is about 5.50 seconds, and that of transaction $\mathsf{Tx}_{\mathsf{Mint}}$ is approximately 6.14 seconds in protocol Θ, as demonstrated in Figure 10 (a). The corresponding network throughput for $\mathsf{Tx}_{\mathsf{Burn}}$ and $\mathsf{Tx}_{\mathsf{Mint}}$ are 45.78 and 32.58, respectively, as shown in Figure 10 (b). In protocol Φ (shown in Figures 10 (c), (d)), the transaction $\mathsf{Tx}_{\mathsf{Lock}}$ takes around 5.44 seconds to complete, while the transaction $\mathsf{Tx}_{\mathsf{Unlock}}$ takes around 6.21 seconds. The corresponding network TPS values for $\mathsf{Tx}_{\mathsf{Lock}}$ and $\mathsf{Tx}_{\mathsf{Unlock}}$ are 48.38 and 32.55, respectively. The increasing number of nodes and transactions is responsible for this trend because the communication time between the nodes and the number of transactions waiting in the queue simultaneously increase, leading to longer transaction delays and decreased network throughput. To mitigate this problem, it is recommended to implement layer-2 scaling solutions [15] to lower the transaction delay and enhance the system throughput.

Finally, we simulate the gas consumption of each protocol in two 20-node blockchains. As shown in Table 2, the protocol

Table 2: Gas consumption of each protocol.

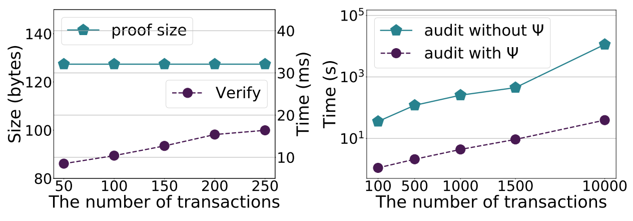
Protocol	Gas	ETH	USD*
Θ	494,000	0.000494	1.72
Φ	901,472	0.000901	3.13

* Gasprice = 1 Gwei, 1 Ether = 10^9 Gwei, and 1 Ether = 3,470.71 USD.

Θ consumes 494,000 gas (equivalent to around 1.72 dollars), while the protocol Φ requires 901,472 gas (about 3.13 dollars). The gas consumption of Θ is mainly due to proof verification, whereas the gas consumption Φ mainly arises from the proof verification and multiple interactions with smart contracts.

7.2.2 Cross-Chain Auditing

In this experiment, we build a 20-node audit chain, to audit whether whether all senders of transactions are included in the blacklist of the audit chain. The experimental results are reported in Figures 11 (a), (b). Specifically, in Figure 11 (a),



(a) (b)

Figure 11: The proof size, proof verification time (left), and audit time (right) of the cross-chain auditing protocol Ψ vary with the number of transactions.

we initially evaluate the impact of the number of aggregated transactions within circuit Λ_Ψ (shown in Figure 8) on both the audit time and the size of a proof. We vary the number of transactions from 50 to 250 and observe an increase in the time required for auditing (proof verification) from 8.52 milliseconds to 16.43 milliseconds, while the proof size remains 127.38 bytes. One can see that, owing to the succinctness of zk-SNARK, the growing number of transactions aggregated within a single proof leads to only a slight increase in verification time, with no change in the size of the proof.

Subsequently, in Figure 11 (b), we conduct a comparative experiment on the audit efficiency with protocol Ψ and full auditing without Ψ. We vary the transaction quantity in the range of 100 to 10,000, which corresponds to the number of ordinary chains ranging from 1 to 100, each being assigned to processing 100 transactions. One can see that the audit time without Ψ is about 35.66 seconds, while with Ψ, it is only 1.10 seconds. However, when the number of transactions is increased to 10,000, the network congestion among the 20 nodes causes the audit time to be around 3.15 hours without Ψ. In contrast, with Ψ, the audit time is decreased to about 40 seconds under the same condition, which demonstrates a significant improvement in efficiency.

Table 3: Gas consumption of the protocol Ψ.

Protocol	Gas	ETH	USD*
Ψ	466,520	0.000467	1.62

* Gasprice = 1 Gwei, 1 Ether = 10^9 Gwei, and 1 Ether = 3,470.71 USD.

Then, based on the experimental setup of gas outlined in Section 7.2.1, we conduct tests to determine the gas consumption required for the auditing protocol Ψ, where each proof contains 50 transactions. As shown in Table 3, the gas required for executing Ψ is 466,520. Note that, as mentioned in Section 5.3, proof verification can be implemented without the use of smart contracts, thus reducing gas consumption.

Table 4: Storage cost of zk-Rollup and our scheme.

Solution	Constraints	Pubic inputs (KB)	Veri. key (KB)
zk-Rollup	11,466,073	10.21	12.20
Our work	11,763,593	**6.83**	**0.24**

Finally, we simulate the storage cost of zk-Rollup (introduced in Section 4.2) and our protocol Ψ, with both circuits containing 100 transactions. As shown in Table 4, Ψ adds circuit constraints but reduces the size of data, i.e., public inputs and verification keys, to be uploaded to the audit chain, thereby reducing the storage cost for auditors. Based on Ethereum's storage cost, storing 256 bits of data costs 20,000 units of gas [5]. Therefore, verifying 100 transactions can conserve approximately 16 KB of space and save about 10 million units of gas. This saving is significant for blockchain networks with billions of transactions.

8 Conclusion

In this paper, we propose zkCross, a cross-chain scheme that supports privacy protection and efficient auditing. zkCross employs a novel auditing architecture with three protocols to address the Cross-chain Linkability Exposure (CLE) problem, the Incompatibility of Privacy and Auditing (IPA), and the Full Auditing Inefficiency (FAI) problem. We conduct a thorough security analysis and carry out comprehensive simulation studies, and our results indicate that zkCross can bring higher audit efficiency while preserving privacy. In our future research, we will focus on enhancing the system's resilience against attacks while maintaining privacy. Moreover, we will extend zkCross to support multi-layer (more than 2) auditing, thereby expanding its application scenarios and further optimizing the audit efficiency.

Acknowledgment

We would like to thank the anonymous USENIX Security shepherd and reviewers for their constructive feedback that helped us improve the paper. This study was partially supported by the National Key R&D Program of China (No.2023YFB2703600), the National Natural Science Foundation of China (No.62232010, 62302266, U23A20302), the Shandong Science Fund for Excellent Young Scholars (No.2023HWYQ-008), and the Shandong Science Fund for Key Fundamental Research Project (ZR2022ZD02).

References

[1] Foteini Baldimtsi, Ian Miers, and Xinyuan Zhang. Anonymous sidechains. In *International Workshop on Data Privacy Management*, pages 262–277. Springer, 2021.

[2] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8):1–41, 2021.

[3] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, pages 486–504. Springer, 2014.

[4] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. Zexe: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 947–964. IEEE, 2020.

[5] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, 2020.

[6] Apoorvaa Deshpande and Maurice Herlihy. Privacy-preserving cross-chain atomic swaps. In *Financial Cryptography and Data Security*, pages 540–549. Springer, 2020.

[7] Yuefeng Du, Huayi Duan, Anxin Zhou, Cong Wang, Man Ho Au, and Qian Wang. Enabling secure and efficient decentralized storage auditing with blockchain. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3038–3054, 2021.

[8] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 106–123. IEEE, 2019.

[9] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 81–98. Springer, 2017.

[10] Alberto Garoffolo, Dmytro Kaidalov, and Roman Oliynykov. Zendoo: A zk-snark verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 1257–1262. IEEE, 2020.

[11] Zhonghui Ge, Jiayuan Gu, Chenke Wang, Yu Long, Xian Xu, and Dawu Gu. Accio: Variable-amount, optimized-unlinkable and nizk-free off-chain payments via hubs. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1541–1555, 2023.

[12] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1259–1273, 2022.

[13] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[14] Yihao Guo, Zhiguo Wan, Hui Cui, Xiuzhen Cheng, and Falko Dressler. Vehicloak: A blockchain-enabled privacy-preserving payment scheme for location-based vehicular services. *IEEE Transactions on Mobile Computing*, (01):1–13, 2022.

[15] Yihao Guo, Minghui Xu, Dongxiao Yu, Yong Yu, Rajiv Ranjan, and Xiuzhen Cheng. Cross-channel: Scalable off-chain channels supporting fair and atomic cross-chain operations. *IEEE Transactions on Computers*, pages 1–14, 2023.

[16] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *Network and distributed system security symposium*, 2017.

[17] Jun Wook Heo, Gowri Sankar Ramachandran, Ali Dorri, and Raja Jurdak. Blockchain data storage optimisations: A comprehensive survey. *ACM Computing Surveys*, 2024.

[18] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pages 245–254, 2018.

[19] Gershon Kedem and Yuriko Ishihara. Brute force attack on {UNIX} passwords with {SIMD} computer. In *8th USENIX Security Symposium (USENIX Security 99)*, 1999.

[20] Peter Kietzmann, Thomas C Schmidt, and Matthias Wählisch. A guideline on pseudorandom number generation (prng) in the iot. *ACM Computing Surveys (CSUR)*, 54(6):1–38, 2021.

[21] Yuxian Li, Jian Weng, Ming Li, Wei Wu, Jiasi Weng, Jia-Nan Liu, and Shun Hu. Zerocross: A sidechain-based privacy-preserving cross-chain solution for monero. *Journal of Parallel and Distributed Computing*, 169:301–316, 2022.

[22] Manlu Liu, Kean Wu, and Jennifer Jie Xu. How will blockchain technology impact auditing and accounting: Permissionless versus permissioned blockchain. *Current Issues in auditing*, 13(2):A19–A29, 2019.

[23] Sinisa Matetic, Karl Wüst, Moritz Schneider, Kari Kostiainen, Ghassan Karame, and Srdjan Capkun. {BITE}: Bitcoin lightweight client privacy using trusted execution. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 783–800, 2019.

[24] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. *Version v0*, 31:15, 2008.

[25] Babu Pillai, Kamanashis Biswas, Zhé Hóu, and Vallipuram Muthukkumarasamy. Burn-to-claim: An asset transfer protocol for blockchain interoperability. *Computer Networks*, 200:108495, 2021.

[26] Xianrui Qin, Shimin Pan, Arash Mirzaei, Zhimei Sui, Oğuzhan Ersoy, Amin Sakzad, Muhammed F Esgin, Joseph K Liu, Jiangshan Yu, and Tsz Hon Yuen. Blindhub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2462–2480. IEEE, 2023.

[27] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[28] Amritraj Singh, Kelly Click, Reza M Parizi, Qi Zhang, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, 149:102471, 2020.

[29] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^{2}l: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1834–1851. IEEE, 2021.

[30] Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, and Pedro Moreno-Sanchez. Universal atomic swaps: Secure exchange of coins across all blockchains. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1299–1316. IEEE, 2022.

[31] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. Mad-htlc: because htlc is crazy-cheap to attack. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1230–1248. IEEE, 2021.

[32] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016.

[33] Xiaoyi Wang, Weiwei Qiu, Lei Zeng, Hongkai Wang, Yiyang Yao, and Dong He. A supervisory and governance mechanism for power master-slave chain architecture. In *2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI)*, pages 172–175. IEEE, 2021.

[34] Jiasi Weng, Jian Weng, Jilian Zhang, Ming Li, Yue Zhang, and Weiqi Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2438–2455, 2019.

[35] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper*, 21:2327–4662, 2016.

[36] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3003–3017, 2022.

[37] Lei Xu, Ting Bao, and Liehuang Zhu. Blockchain empowered differentially private and auditable data publishing in industrial iot. *IEEE Transactions on Industrial Informatics*, 17(11):7659–7668, 2020.

[38] Minghui Xu, Yihao Guo, Chunchi Liu, Qin Hu, Dongxiao Yu, Zehui Xiong, Dusit Niyato, and Xiuzhen Cheng. Exploring blockchain technology through a modular lens: A survey. *arXiv preprint arXiv:2304.08283*, 2023.

[39] Longyang Yi, Yangyang Sun, Bin Wang, Li Duan, Hongliang Ma, Bin Wang, Zhen Han, and Wei Wang. Ccubi: A cross-chain based premium competition scheme with privacy preservation for usage-based insurance. *International Journal of Intelligent Systems*, 37(12):11522–11546, 2022.

[40] Zeyuan Yin, Bingsheng Zhang, Jingzhong Xu, Kaiyu Lu, and Kui Ren. Bool network: An open, distributed, secure cross-chain notary platform. *IEEE Transactions on Information Forensics and Security*, 17:3465–3478, 2022.

[41] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William Knottenbelt. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 193–210. IEEE, 2019.

[42] Shijie Zhang and Jong-Hyouk Lee. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE transactions on Industrial Informatics*, 15(10):5715–5722, 2019.

[43] Xiaoyan Zhang, Jingwei Chen, Yong Zhou, and Shunrong Jiang. Privacy-preserving cross-chain payment scheme for blockchain-enabled energy trading. In *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 109–114. IEEE, 2021.

[44] Yushu Zhang, Jiajia Jiang, Xuewen Dong, Liangmin Wang, and Yong Xiang. Bedcv: Blockchain-enabled decentralized consistency verification for cross-chain calculation. *IEEE Transactions on Cloud Computing*, 2022.

[45] Qingyi Zhu, Seng W Loke, Rolando Trujillo-Rasua, Frank Jiang, and Yong Xiang. Applications of distributed ledger technologies to the internet of things: A survey. *ACM computing surveys (CSUR)*, 52(6):1–34, 2019.