CrossChannel: Efficient and Scalable Cross-Chain Transactions Through Cross-and-Off-Blockchain Micropayment Channel

Xinyi Luo, Kaiping Xue, *Senior Member, IEEE,* Qibin Sun, *Fellow, IEEE*, Jun Lu

***Abstract*—The surge in blockchain-based cryptocurrencies has created a pressing need for Cross-Chain Transaction (CCTx) solutions. Existing solutions either lack sufficient security, like centralized exchanges, or suffer from poor efficiency and scalability, such as atomic swaps. Inspired by the success of the Lightning Network in accelerating Bitcoin transactions, we propose CrossChannel that establishes cross-and-off-chain micropayment channels to achieve efficient and scalable CCTx. Specifically, we analyze the challenges of extending one-chain channels to cross-chain scenarios caused by the separation of blockchains. To overcome these challenges, we employ the chain relay mechanism to synchronize channel-related information across blockchains and construct the channel management protocol on this basis, ensuring the same security level as one-chain channels in cross-chain settings. We prototype CrossChannel between two Ethereum testnets, comparing its transaction efficiency and costs with a typical HTLC-swap scheme. Results demonstrate the significant advancements in efficiency and scalability offered by CrossChannel. Even with channels closing after just 20 transactions, CrossChannel exhibits a fivefold capacity increase for handling CCTxs compared to HTLC swaps.**

***Index Terms*—blockchain, cross-chain transaction, payment channel, scalability, off-chain payment**

I. Introduction

The emergence of blockchain-based cryptocurrencies has emphasized the need for Cross-Chain Transactions (CCTx) [1]. For instance, if Alice exclusively holds bitcoins but needs to pay Bob in ethers, a Bitcoin-to-Ethereum transaction becomes necessary. CCTx has garnered significant attention and research efforts from academia and industry alike. Initially, centralized exchanges were commonly utilized, where a trusted intermediary facilitated cryptocurrency exchanges. However, these centralized exchanges are susceptible to theft of traders' funds [2], [3]. For example, Mt. Gox, formerly one of the leading global bitcoin exchanges, announced the disappearance of approximately $850,000$ bitcoins valued at around $\$480$ million at that time, believed to be stolen, ultimately resulting in bankruptcy [2].

Given the growing concerns surrounding centralized exchanges, researchers are exploring decentralized solutions for CCTx to address the vulnerabilities introduced by trusted intermediaries. One of the most well-known solutions is the atomic swap scheme, which is based on Hash Timelock Contracts (HTLC) [4]–[6]. HTLC allows a user to lock assets in a contract, and another user can withdraw those assets by providing the required secret. If the secret is not provided within a specified time limit, the original user can withdraw their assets. For instance, Alice on the Bitcoin blockchain and Bob on the Ethereum blockchain can conduct an HTLC-swap with the assistance of Carol, who has accounts on both blockchains and may be malicious. The HTLC-swap typically involves at least four steps. First, Alice locks her bitcoins in the HTLC_{BTC}. Then, Carol locks her ethers in the HTLC_{ETH}. Next, Bob submits a secret to HTLC_{ETH} to acquire the locked ethers. Finally, Carol submits the secret to HTLC_{BTC} to withdraw the locked bitcoins.

The process of atomic swaps is lengthy. It involves multiple rounds of confirmations on different blockchains, significantly slowing down transaction processing. Furthermore, these multiple confirmations result in users having to pay transaction fees to blockchain miners multiple times. Additionally, users are required to pay fees to the intermediary, such as Carol, making CCTx extremely costly. While there are alternative solutions, such as notarized atomic swaps [7], [8] and sidechains [9]–[11], the issues of low efficiency and high expenses persist. Taking into account these challenges and drawing inspiration from the success of the Lightning Network [12] in facilitating Bitcoin transactions, we propose the establishment of cross-and-off-chain payment channels.

The core concept of the Lightning Network is to allow two users to create an off-chain payment channel by locking assets on the blockchain. Subsequently, they can conduct transactions without directly broadcasting them to the blockchain. Instead, they sign their balance as a payment proof for each transaction. Let's consider an example where Alice locks x assets and Bob locks y assets, resulting in an initial balance of (x, y). And after the n^{th} payment, their balance becomes (x_n, y_n). If they choose to close the channel at this point, either user can submit (x_n, y_n) along with the two signatures to the blockchain. The blockchain miners verify the correctness of the balance, ensuring that the total balance x_n+y_n equals $x+y$, and verify the correctness of the signatures. If both valid, the locked assets will be transferred to the respective users based on the balance. In the event that an outdated balance is submitted, the payment channel provides a dispute mechanism that allows the other participant to present a more recent one. By utilizing off-chain payment channels, only two transactions (one for channel creation and another for channel closing) are required

X. Luo, K. Xue, Q. Sun and J. Lu are with the school of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China.

J. Lu is also with the Department of Electronic Engineering and Information Science, Hefei, Anhui 230027, China

Corresponding Author: K. Xue (E-mail: kpxue@ustc.edu.cn)

to support numerous payments. This significantly increases the overall transaction throughput and allows users to benefit from lower transaction fees.

Extending a one-chain payment channel to cross-chain scenarios presents several security issues, referred to as the "colluded deception" and "inability to dispute". First, neither blockchain can verify the total locked balance since users can only lock assets on their respective blockchains independently. This creates an opportunity for collusion between the two users, allowing them to deceive the blockchains regarding the balance. Consequently, they can acquire more assets during channel closure than the amount they have actually locked. Second, even if one user is honest, the channel may still be insecure due to the lack of dispute capability. This is because a user cannot send transactions without an account, thus cannot dispute. Therefore, if a user attempts to close the channel with an outdated balance, the blockchain cannot identify the problem through notifications from the other user.

Both the aforementioned issues, including colluded deception and inability to dispute, arise due to the separation of blockchains. To address these challenges, we employ a chain relay mechanism to synchronize channel-related information across blockchains. The chain relay serves two primary purposes. First, it synchronizes channel snapshots during channel creation, enabling the two blockchains to maintain a consistent view of the channel, particularly with respect to confirming the total locked balance. Second, it synchronizes balance proofs and disputes submitted by users during channel closure, thereby facilitating mutual supervision among users. In addition to the security concerns, the cross-chain channel also faces the issue of asset availability. For instance, if the settled assets exceed the locked assets, there may not be sufficient assets available for settlement. To mitigate this problem, we propose the adoption of a bank-based asset management mechanism. In summary, the primary contributions of this work are as follows:

- We propose utilizing off-chain payment channels to facilitate cross-chain transactions. To address the security and availability challenges caused by the separation between blockchains, we introduce a chain relay protocol and a bank-based asset management mechanism. As a result, we present CrossChannel that enables two users on different blockchains to establish an off-chain payment channel for conducting fast and cheap cross-chain payments.
- To ensure the efficiency and security of the chain relay, we design the channel snapshot mechanism that contains all channel-related information within a short content, acting as the relayed content. In addition, we employ a Proof-of-Stake (PoS) consensus protocol adopted by relayers, guaranteeing the correctness of chain relay.
- We prove that the cross-chain channel achieves the same security level as that of one-chain channels. Besides, we prototype CrossChannel between two Ethereum-based testnets to evaluate its practical overhead. We use the queuing theorem to systematically analyze CrossChannel's performance and compare it with the basic HTLC-swap scheme. Results show that, even when each channel undertakes only 20 transactions, CrossChannel still realizes a fivefold performance compared to HTLC swap.

The rest of this paper is organized as follows. Section II presents the related work. Section III introduces the preliminaries of blockchains. In Section IV, we first show our insights of establishing cross-chain payment channels, analyzing the issues and solving ideas (IV-A). On this basis, Section IV-B to IV-D describe the system model, security assumptions, and design goals respectively. Section V elaborates the proposed solution by three steps, i.e., defining the channel data structure and snapshot mechanism (V-A), introducing channel management protocol (V-B), and explaining the chain relay mechanism (V-C). Sections VI and VII give the security and performance analysis. Section VIII concludes the paper.

II. Related Work

A classic solution for Cross-Chain Transactions (CCTx) is the atomic swap scheme based on the Hash Timelock Contract (HTLC). The process of HTLC swaps can be summarized as follows. When Alice on blockchain A needs to pay Bob on blockchain B, she finds another user, Carol, who has accounts on both blockchains. Then it takes at least four steps to complete the payment: Alice locks x coin^A in HTLC on blockchain A, Carol locks y coin^B in HTLC on blockchain B, Bob withdraws the y coin^B from HTLC on blockchain B, and Carol withdraws the x coin^A from HTLC on blockchain A. It was first introduced in [13] and formalized in [4]. Subsequently, researchers optimized HTLC swaps in terms of efficiency [14], security [5], [6], and usability in scriptless blockchains [15]. While these cryptographic-based atomic swaps provide sufficient security, they also have some notable flaws, such as high latency, frequent interactions, and expensive transaction fees. As a result, HTLC atomic swaps are not efficient and economic when being put into practical use.

Beyond HTLC-based approaches, other CCTx constructions have emerged, seeking improved efficiency and security. For instance, Interledger [7] utilizes notary nodes running PBFT consensus to validate lock/withdraw processes. Tesseract [8] leverages trusted execution environments for enhanced privacy and throughput. While advancing performance in various ways, these approaches have yet to achieve a step-change in scalability. Additionally, sidechains [9]–[11], [16], [17] connecting two ledgers through coin locking/burning and reissuance represent an important CCTx paradigm. However, these constructions cannot provide solutions for classic CCTx, i.e., transactions between users on different blockchains.

Despite all the efforts, existing CCTx solutions still lack the efficiency required for practical use, mainly due to cross-chain synchronization demands. To solve the issue, we turn to off-chain payment channels as a potential solution. Off-chain payment channels are proposed by the Lightning Network [12] to address the scalability problem of Bitcoin. Since then, researchers have conducted many studies on enhancing privacy [18]–[20] and performance [21], [22] of payment channels. With off-chain payment channels, users can complete transactions through the channel without notifying the blockchains for each transaction. Instead, multiple transactions can be submitted with just one confirmation, significantly improving scalability and reducing the burden on the blockchain.

Several studies have tried to improve the scalability of CCTx utilizing off-chain payment channels. For instance, the Arwen protocol [23] designs a cryptocurrency exchange protocol for users to exchange assets. Arwen sets a centralized exchange for each blockchain. When a user needs to exchange assets between two blockchains, he/she can establish an off-chain payment channel with a centralized exchange on either blockchain. The payment channel records the balance of both assets simultaneously. In [24], a cryptocurrency exchange protocol is proposed based on the Lightning Network. Unlike Arwen, this protocol constructs a pair of payment channels on two blockchains for one exchange and does not need a centralized exchange. In conclusion, those existing studies only provide solutions for cross-chain exchanges where a user has accounts on both blockchains. Thus, establishing off-chain payment channels between two users located on different blockchains remains unsolved.

III. Preliminaries

A. Off-Chain Payment Channel

To improve efficiency of blockchain transactions, Poon *et al.* [12] proposed the Lightning Network, which introduces the concept of off-chain payment channels and implements it on Bitcoin. Subsequently, the Raiden Network [25] implements payment channels on Ethereum. We take the Raiden Network as an example to introduce the off-chain payment channel, because it is implemented based on smart contracts and is easier to understand. Notably, both the Lightning Network and Raiden Network share the same underlying principles. However, since Bitcoin only supports simple scripts, the specific implementation of the Lightning Network is more complex.

Fig. 1(a) shows an example of a payment channel. Assuming Alice and Bob decide to create a payment channel, they need to lock some assets in the Channel Management Contract (CMC). Subsequently, they can conduct off-chain payments. For each payment, they sign the up-to-date balance to generate a balance proof. In the given example, the balance proof for the first payment is denoted as $\pi_1 = (\langle 3, 3\rangle, \sigma_A, \sigma_B)$. When either user decides to close the channel, he/she is supposed to submit the final balance proof to the CMC. The CMC verifies the two signatures and checks if the total balance equals the locked amount. If everything is valid, the CMC transfers the locked assets to Alice and Bob according to the balance. However, a challenge arises as the CMC lacks a means to validate the freshness of the balance proof. In the given example, Alice may submit π_1 to close the channel because her balance in π_1 is larger than that in π_2. Therefore, the payment channel provides a dispute mechanism by waiting for a certain time period to allow Bob to submit a fresher balance proof.

The security of a payment channel relies on several aspects, namely balance proofs, the fixed balance, and dispute mechanism. First, after each payment, both users sign the up-to-date balance to form the balance proof, ensuring the non-repudiation of off-chain payments. Second, the total balance in the channel is fixed, guaranteeing that when the channel is closed, the assets transferred to the users are equivalent to their initial locked amount. Lastly, the dispute mechanism prevents dishonest users from submitting outdated balances. As a result, once an off-chain payment is conducted, the blockchain will accept it when the channel is closed.

B. Chain Relay

Chain relay is an important technology that enables blockchain interoperability. It was first proposed as BTC relay [26], which records Bitcoin block headers in Ethereum, allowing users to verify Bitcoin transactions on Ethereum. The main concept behind chain relay is to design succinct proofs for the information that needs to be verified and select a group of relayers to upload these proofs to the targeted blockchain. There are three key aspects to consider in chain relay:

- *Proofs:* The design of the proof depends on the specific content that needs to be verified. It is crucial for the proof to be succinct in order to minimize the overhead on the blockchains when conducting chain relay. For instance, Westerkamp *et al.* [27] proposed a batch-aggregating zero-knowledge proof-based relay system, which significantly reduces the size of proofs compared to BTC relay.
- *Correctness:* Correctness is a vital aspect of chain relay, ensuring that only valid proofs are relayed. This is typically achieved by employing consensus protocols among the relayers, similar to how blockchain miners operate.
- *Incentives:* Incentives play a crucial role in encouraging desirable behavior from relayers and ensuring the healthy operation of chain relay. Essentially, relayers who provide correct proofs are rewarded, while those who engage in malicious activities should face appropriate penalties.

C. PoS Consensus

There are several essential and frequently-used consensus mechanisms, such as Proof-of-Work (PoW) [28], Proof-of-Stake (PoS) [29], Delegated Proof-of-Stake (DPoS) [30], Practical Byzantine Fault Tolerance (PBFT) [31], and so on.

Among them, PoW requires massive computational costs, which can be extremely expensive. PBFT requires additional incentive methods to award honest nodes or identify and punish malicious nodes. On the other hand, both PoS and DPoS assign an attribute called "stake" to each node, which can be leveraged for incentives. For example, the Ouroboros [29] protocol randomly selects a node to act as the leader for generating a new block. The probability of a node being selected is proportional to its stake, implying that the reward for generating blocks is related to the stake. DPoS [30] enables a node to delegate its stake to another node, allowing it to not directly participate in the consensus. Instead, the delegate acquires the delegated stake and has a stronger impact on block generation. The delegate mechanism reduces consensus costs by reducing the number of nodes that practically participate in the consensus. For the same purpose, Ethereum [32] adopts the committee mechanism, which requires a blockchain node to deposit 32 Ethers to become a validator and participate in the block generation process. If a validator is discovered to be malicious, it will lose all the deposited Ethers. During consensus, a group of validators is selected to form the committee, and a leader is randomly chosen from the committee to generate

a new block, while other nodes in the committee vote on the block to validate it. Considering the incentive ability of PoS, we adopt the PoS consensus for chain relay.

IV. Overview of Cross-Chain Payment Channel

In this section, we first demonstrate the issues of extending one-chain payment channels to cross-chain scenarios. We conclude three key issues and introduce our main idea to solve them (IV-A). Next, we present the system model (IV-B), security assumptions (IV-C), and design goals (IV-D) of CrossChannel.

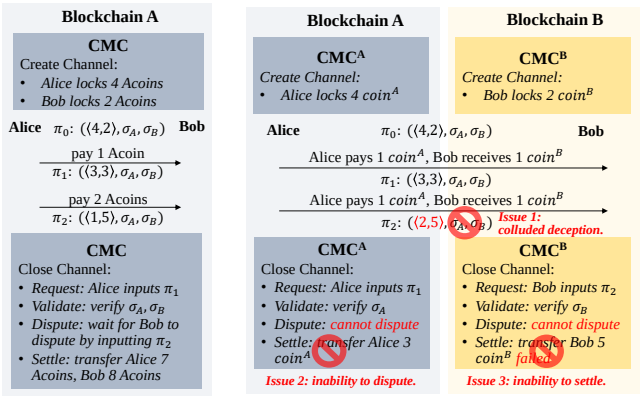
A. Insights and Challenges

Intuitively, to establish a cross-chain payment channel, the Contract Management Contract (CMC) is divided into two parts, separately deployed on the two blockchains, denoted as CMC^A and CMC^B, as depicted in Fig. 1(b). Accordingly, Alice and Bob lock their assets to CMC^A and CMC^B respectively to create a payment channel. They can then conduct off-chain payments and, finally, submit the balance proof to the two CMCs to close the channel. However, there are several issues that make the channel insecure or unavailable.

- *Issue 1: colluded deception.* During channel creation, CMCs cannot verify the assets locked in each other. Consequently, they cannot confirm the total balance in the channel. This creates an opportunity for the two users to generate balance proofs in which their total balance exceeds their locked amount. As a result, their settled assets end up being more than the locked assets. We refer to this issue as the colluded deception issue, as the two users collude to deceive assets from blockchains.
- *Issue 2: inability to dispute.* Even if the balance is correct, the security of the channel is compromised because dispute becomes impossible. This is because Bob cannot invoke CMC^A due to a lack of an account on blockchain A. As a result, when Alice submits an old balance proof, CMC^A can only settle Alice accordingly.
- *Issue 3: inability to settle.* In the given case depicted in Fig. 1(b), CMC^B cannot settle Bob because Bob locked only 2 coin^B, while he should receive 5. On the contrary, several coin^A remain in CMC^A.

Overall, issues 1 and 2 arise due to the separation of blockchains, while issue 3 is caused by the transfer of assets among blockchains. To solve these issues, we leverage chain relay technology to synchronize key information between the two CMCs. This enables balance confirmation and cross-chain dispute resolution, addressing issues 1 and 2. Additionally, we address issue 3 by designing a bank-based settlement mechanism. When CMC closes a channel in which the user corresponding to the current blockchain is the payer, the redundant assets will be transferred to the bank. Those assets can be used to settle other channels where the user on the current blockchain is the payee. To ensure asset availability of the bank, relayers are required to deposit some assets when registering, and the deposit is leveraged as reserves.

To apply chain relay appropriately and securely, we design a channel snapshot mechanism that compresses all information related to a channel into a concise snapshot, which is the content to be relayed. Then, we adopt the Proof-of-Stake (PoS) consensus algorithm among relayers. The PoS consensus is supported by a stake-based incentive framework, ensuring security and liveness of the relay process.



(a) one-chain channel (b) cross-chain channel

Fig. 1: Intuition of cross-chain payment channel.

B. System Model

Combining the solutions for the three issues, we propose the CrossChannel protocol that enables users on different blockchains to establish secure cross-chain payment channels. Fig. 2 illustrates the overall system model of CrossChannel. Regardless of the specifics of the underlying blockchains, CrossChannel consists of two types of entities: users with a significant demand for cross-chain micropayments and relayers who contribute to the chain relay process to earn rewards.

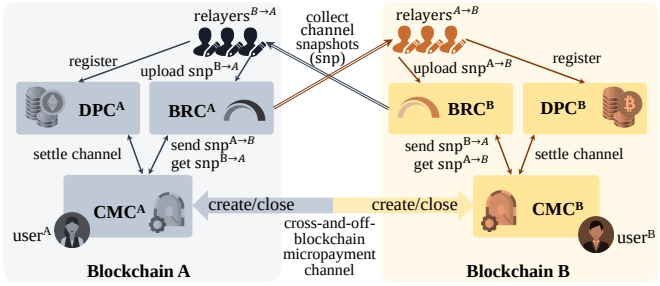


Fig. 2: The system model of CrossChannel.

- **Users** belong to different blockchains and do not have accounts on others' blockchains due to a lack of permission, capability, or willingness. This leads to a demand for cross-chain transactions. Users are willing to pay fees to create an off-chain payment channel, enabling them to conduct fast and cost-effective cross-chain transactions.
- **Relayers** are blockchain users who contribute to the chain relay process to earn rewards. For example, a user of blockchain A who has access to blockchain B can register as a $\text{relayer}^{B\to A}$ by depositing some assets on blockchain A. The superscript indicates the destination of the relayer, and a $\text{relayer}^{B\to A}$ collects the required relay contents on blockchain B and records them on blockchain A. Notably,

having access to blockchain B does not require having an account on blockchain B; it only requires the ability to read blocks on blockchain B.

Each blockchain provides three functionalities to facilitate the entities finishing their tasks. The three functionalities are implemented via smart contracts, including the Channel Management Contract (CMC), Bridge Contract (BRC), and Deposit Contract (DPC). Among them, the CMC manages channel creation and closure procedures, maintaining the channel-related information. The two CMCs related to the same channel ensure their consistency with the help of BRCs. The BRC synchronizes channel snapshots between the related CMCs through chain relay. It utilizes a Proof-of-Stake (PoS) consensus mechanism to validate the correctness of relayed contents submitted by relayers. Finally, the DPC serves as the bank used for channel settlement.

C. Security Assumptions

There are three factors that can impact security: the underlying blockchains, the channel management protocol, and the chain relay mechanism. First, since the proposed CrossChannel protocol does not focus on the underlying blockchains, we assume they are secure. Additionally, exchange rates between blockchains are publicly acknowledged. Second, regarding the channel management protocol, as discussed in Section IV-A, there are two potential malicious behaviors of channel users, including colluding to attack the blockchains (issue 1 of Section IV-A) or engaging in a mutual attack against each other (issue 2 of Section IV-A).

Finally, concerning the chain relay mechanism, there are two potential negative behaviors that relayers may exhibit: lazy relayers, who participate in the relay process less frequently, and cautious relayers, who are hesitant to leave their assets in the DPC, which may negatively impact the availability of the bank-based settlement mechanism. We assume that the majority of relayers are profit-driven and rational. They participate in the process to earn rewards and will take actions that benefit themselves the most. Additionally, a minority of relayers may collude with malicious users, attempting to include incorrect channel information in the relay contents. This could potentially facilitate attacks by users. We assume that the proportion of malicious relayers is denoted by ρ, and ρ must be less than $\frac{1}{2}$. If ρ reaches $\frac{1}{2}$, the security of the proposed scheme is compromised.

D. Design Goals

The primary purpose of CrossChannel is to propose a secure cross-chain payment channel protocol that enables users to conduct fast and cost-effective cross-chain transactions. This primary goal can be divided into two steps. First, assuming the existence of a reliable chain relay mechanism that synchronizes channel snapshots among blockchains, our aim is to design a secure channel management protocol that can effectively address the three issues discussed in Section IV-A.

Second, we aim to design the chain relay mechanism required by the channel management protocol. To achieve this, it is crucial to adopt an appropriate consensus mechanism among the relayers, ensuring the correctness of relayed contents with the existence of a certain proportion (ρ) of malicious relayers. Besides, a proper incentive mechanism is required to ensure the security and liveness of the relay process. According to [33], an incentive mechanism should adhere to several principles: a) Effectiveness: It should accurately identify honest relayers, rewarding them for their honesty, while also penalizing malicious ones. Additionally, it should be able to prevent rational relayers from being lazy or cautious. b) Sustainability: The mechanism should prevent any imbalances among system participants, such as resource monopolies, ensuring the long-term operation of the system. c) Fairness: The incentive mechanism should be executed reliably, without the possibility of manipulation.

V. Proposed Solution

This section describe the CrossChannel protocol with four steps. First, Section V-A defines the Channel data structure and introduce the channel snapshot. Section V-B illustrates the lifecycle of a cross-chain channel, including channel creation, off-chain payment, and channel closure. In this part, the details of the chain relay protocol are omitted and abstracted as two interfaces, $\mathrm{RelayReq}$ and $\mathrm{RelayGet}$ which can be invoked by CMC. Then, Section V-C provides a detailed description of the chain relay protocol. Finally, Section V-D explains how to extend CrossChannel to contractless blockchains.

A. Definition of the Channel

Before introducing the CrossChannel protocol, we first define the data structure of a cross-chain payment channel, denoted as Channel. As shown in Fig. 3, a Channel consists of two parts: channel_info and channel_snp.

- channel_info represents the basic information of the channel, including the blockchains (chainpair$=(A, B)$), the addresses of the two users (addrpair$=(\mathsf{addr}_1^A, \mathsf{addr}_2^B)$), the initial balances locked by the users (balpair$=(x, y)$), the exchange rate between the blockchains (γ), and the dispute time τ_d. This information is provided by the user when requesting to create the channel.
- channel_snp is the snapshot of a channel and contains the channel identifier cid and the channel state state. cid serves as a unique identifier for the channel, which is the hash of channel_info. state is used to indicate the different stages in the lifecycle of a channel. There are four possible states, namely init, activated, preclosed[bp], and closed[bp]. The explanations of these states are provided below.

When intending to create a channel, users should negotiate the parameters of channel_info and submit the same information to their respective CMCs. They should also lock assets according to the balpair, e.g., U_1^A should lock x coin^A to CMC^A. CMC^A and CMC^B should separately check the locked assets of U_1^A and U_2^B, and initiate the channel with state init only when the assets are correctly locked. Next, the two CMCs can exchange their channel_snp through chain relay (i.e., BRC) to ensure consistency and decide whether to create

Channel

channel_info: the basic information of a channel, including:	
chainpair	The two blockchains of the channel, e.g., chainpair = (A, B).
addrpair	The addresses of the two users, e.g., addrpair = $(\mathsf{addr}_1^A, \mathsf{addr}_2^B)$.
balpair	The two users' locked assets, e.g., balpair = (x, y).
γ	The current exchange rate between the blockchains, i.e., 1 $\mathsf{coin}^A = \gamma\ \mathsf{coin}^B$.
τ_d	The dispute time consulted by the two users.
channel_snp: the snapshot of the channel, including:	
cid	A unique identifier to identify the channel. It is the hash of channel_info, i.e., cid = $\mathrm{Hash}(A\|\|B\|\|\mathsf{addr}_1^A\|\|\mathsf{addr}_2^B\|\|x\|\|y\|\|\gamma\|\|\tau_d)$.
state	The channel state: init, activated, preclose[bp], and closed[bp].
init	The local user has requested for channel creation and locked the initial balance, and is waiting for confirmation from the other CMC.
activated	The channel has been successfully created, users can conduct off-chain payments through it.
preclose[bp]	The local user has submitted a balance proof bp as a request for closing the channel, and is waiting for response from the other CMC.
closed[bp]	The channel has been successfully closed with a balance proof bp.

Fig. 3: The *cross-channel*, denoted as Channel.

the channel. If they acquire a channel_snp with the same cid and state init, they can adjust the state to activated, denoting the success of channel creation. However, if users submit different channel_info or lock incorrect assets, both CMCs will not receive the required channel_snp and will refuse to create the channel. Thus, either both CMCs refuse to create the channel or create an activated channel with consistency.

When a user, e.g., U_1^A, submits an effective balance proof to close the channel, CMC^A changes the state to preclose[bp] and relays it to CMC^B. Subsequently, U_2^B can acquire bp from CMC^B and submit a dispute bp' if it is outdated. If no effective dispute is submitted, CMC^B directly closes the channel with bp and changes the state to close[bp]; otherwise, it sets the state as preclose[bp']. When the channel_snp is relayed to CMC^A, CMC^A acts according to the contained state. In conclusion, there are three possible results of channel closure: a) both CMCs refuse the request and the states remain activated; b) both CMCs close the channel with bp; and c) both CMCs close the channel with bp'. For more details on the channel management protocol, please refer to Section V-B.

B. Channel Management Protocol

The lifecycle of a payment channel consists of three phases: channel creation, off-chain payment, and channel closure. Among them, the channel creation and channel closure phases are managed by the channel management protocol, implemented by the CMC. Users invoke CMCs to create a channel. Then they can conduct off-chain payments through the channel, and finally close it by invoking CMCs. Fig. 4 depicts the lifecycle of a channel, and the detailed explanations are given below.

1) (Phase 1) Channel Creation: Channel creation process consists of two stages: create request ($\mathrm{CreateRequest}$ ①) and channel match ($\mathrm{CreateRequest}$ ②). Suppose two users, U_1^A on $\mathcal{BC}^A$ and U_2^B on $\mathcal{BC}^B$, decide to create a cross-chain payment channel to support their cross-chain transactions. First, during the create request stage, both users invoke the CMC on their respective blockchains to request channel creation. Taking U_1^A as an example, he/she invokes $\mathrm{CreateRequest}$ in CMC^A, locking initial assets and inputting channel_info. Then, CMC^A checks if the locked amount matches the value of x in balpair and verifies the correctness of the exchange rate (γ). If everything is valid, it computes cid and sets state to init. CMC^A then invokes $\mathrm{RelayReq}$ of BRC^A to relay channel_snp to CMC^B. For clarity, we denote the channel_snp generated by CMC^A as snp_1^A, and the channel_snp generated by CMC^B as snp_2^B.

Subsequently, CMC^A enters the channel match stage. After waiting for some time (according to the chain relay frequency), it invokes $\mathrm{RelayGet}(\mathsf{cid})$ of BRC^A to obtain snp_2^B. If U_2^B has successfully invoked $\mathrm{CreateRequest}$ of BRC^B with consistent channel_info, snp_1^A will be equal to snp_2^B, and CMC^A can set state to activated, indicating the successful creation of the channel. Otherwise, the creation fails because the two users did not provide matching information.

2) (Phase 2) Off-chain Payment: After the creation of a payment channel, U_1^A and U_2^B have the ability to conduct off-chain payments by generating balance proofs. These balance proofs, similar to those used in single-chain payment channels, consist of the current balance and the signatures of both users. Furthermore, a txid is included, serving as a sequence number for the transaction and aiding in establishing the order of the balance proofs.

When U_1^A and U_2^B successfully create a payment channel, they generate a bp for the initial balance with $\mathsf{txid} = 0$, i.e.,

$$\mathsf{bp}^0 \leftarrow \langle \mathsf{cid}, 0, (\mathsf{addr}_1^A, \mathsf{addr}_2^B), (x, y), (\sigma_1^A, \sigma_2^B) \rangle.$$

Suppose that after the i-th payment, the balances of U_1^A and U_2^B are x_i and y_i separately. Given that at the $(i{+}1)$-th payment, U_1^A pays m coin^A to U_2^B. Since the exchange rate is γ, after the payment, the balances change to $x_{i+1}{=}x_i{-}m$, $y_{i+1}{=}y_i{+}\gamma m$. Thus, the balance proof is

$$\mathsf{bp}^{i+1} \leftarrow \langle \mathsf{cid}, i{+}1, \mathsf{addr}_1^A, \mathsf{addr}_2^B, (x_{i+1}, y_{i+1}), (\sigma_1^A, \sigma_2^B) \rangle.$$

3) (Phase 3) Channel Closure: Either user of the channel can request to close it by submitting the latest balance proof to CMC. The channel closure procedure consists of two functions: $\mathrm{CloseRequest}$ invoked by the requester and $\mathrm{CloseDispute}$ invoked by another user to dispute. Given that U_1^A is the requester, and U_2^B is the disputer, they take three or four steps to close the channel, as follows.

First ($\mathrm{CloseRequest}$ ①), to close the channel, U_1^A invokes $\mathrm{CloseRequest}$ $(\mathsf{cid}, \mathsf{bp})$ in CMC^A. Then, CMC^A verifies two items: a) whether the balance obeys the exchange rate, i.e., $\delta_x = \gamma\delta_y$, where δ_x and δ_y are the variation of the two users balances; and b) the validity of σ_1^A. If both are correct, CMC^A then sets state to preclose[bp]. Then, it relays $\mathsf{snp}{=}\langle \mathsf{cid}, \mathsf{preclose[bp]} \rangle$ to CMC^B. Notably, the verification of the two signatures is separately taken by the two CMCs. This is because different blockchains may adopt different cryptographic suites, and consortium blockchains may deploy their own Public Key Infrastructure (PKI). Therefore, verifying U_2^B's signature in CMC^A may be unavailable or insecure.

Second ($\mathrm{CloseDispute}$ ②), U_2^B acquires snp_1^A from BRC^B and checks whether bp is the latest one. If it is, U_2^B invokes $\mathrm{CloseDispute}$ in CMC^B inputting nothing; otherwise, he/she invokes it inputting a new balance proof bp'. Upon

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3411820

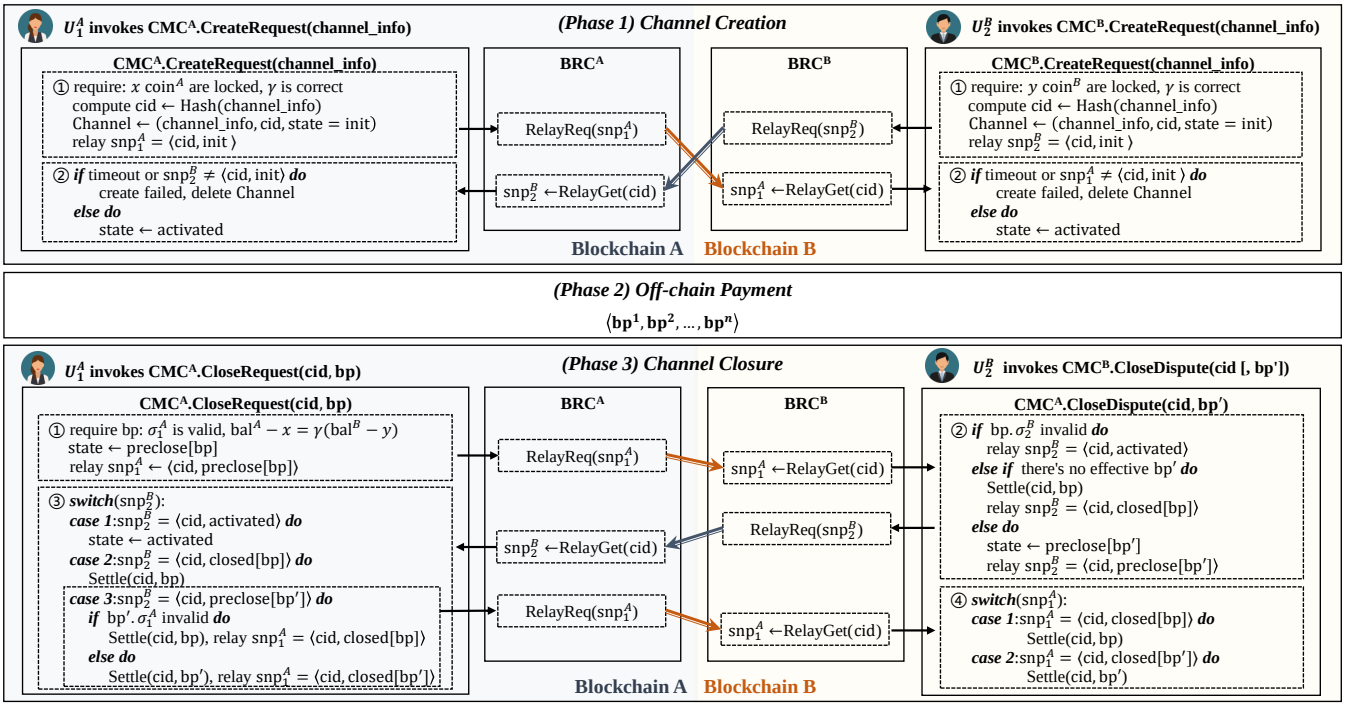


Fig. 4: Life cycle of a *cross-channel*.

CloseDispute being invoked, CMC^B verifies two items: 1) the validity of σ_2^B in bp, and 2) the effectiveness of bp$'$. The verification of bp$'$ is similar to CMC^A's verification of bp, except that bp$'$ should have a large txid compared to bp, indicating its fresher. To conclude, there are three possible verification results, leading to three state values of snp_2^B which will be relayed to CMC^A.

- Case 1: σ_2^B of bp is invalid, indicating that U_1^A submitted an invalid bp. In this case, the request should be rejected. Whatever U_2^B inputs, state remains activated.
- Case 2: U_2^B does not submit an effective dispute. In this case, CMC^B invokes Settle(cid, bp) to close the channel. As a result, state is changed to closed[bp].
- Case 3: U_2^Bsubmits an effective dispute. In this case, CMC^B sets state to preclose[bp$'$].

Third (CloseRequest ③), after CloseDispute returns a snp_2^B, CMC^A can determine the final result of channel closure. According to the three possible state values of snp_2^B, CMC^A takes different actions, as follows:

- Case 1: activated. This implies that CMC^B rejected the request, thus, CMC^A reverses the state to activated.
- Case 2: closed[bp]. This implies that CMC^B closes the channel with bp. In response, CMC^A also closes the channel with bp.
- Case 3: preclose[bp$'$]. This implies that U_2^B submits an effective dispute bp$'$. Then, CMC^A verifies σ_1^A of bp$'$. If it is valid, CMC^A closes the channel based on bp$'$; otherwise, it closes the channel based on bp.

For case 1 and case 2, CMC^A has no need to relay its snapshot, since they can reach consistency immediately. As a result, they take three steps to close the channel or reject the request. However, for case 3, CMC^A should relay the snapshot so that CMC^B can determine which balance proof should be used to close the channel, based on CMC^A's choice (CloseDispute ④). In this case, they take four steps to close the channel.

C. Chain Relay Protocol

The chain relay protocol is used to synchronize channel-related information, i.e., the channel snapshot, consisting of cid, state, and bp, between blockchains when creating and closing payment channels. Fig. 5 illustrates the workflow of relayers$^{A\to B}$ in relaying $\mathsf{snp}^{A\to B}$ from blockchain A to blockchain B. Recall that CMC^A invokes the RelayReq interface to send $\mathsf{snp}^{A\to B}$ to BRC^A. Upon receiving $\mathsf{snp}^{A\to B}$, BRC^A temporarily stores them. Subsequently, relayers$^{A\to B}$ can retrieve all the stored $\mathsf{snp}^{A\to B}$ through the Collect interface, packaging them into a set denoted as $\mathsf{record}^{A\to B}$. They then execute the relay consensus protocol, generate an aggregated signature (σ), and utilize the Upload interface to submit the record and signature to BRC^B. BRC^B can verify the validity of $\mathsf{record}^{A\to B}$ based on σ. At this point, the relay process is completed, and CMC^B can invoke the RelayGet interface of BRC^B to retrieve the required snp.

As Fig. 5 shows, there are four main functions in a BRC: RecordVerification that verifies the records uploaded by relayers, CommitteeSelection and LeaderSelection that are used to determine the relayers participating in the current relay process, and StakeUpdate that computes stakes for relayers. The detailed descriptions are given below.

1) Relay Consensus: Relayers execute a Proof-of-Stake (PoS) consensus protocol to validate the relayed records. In this PoS consensus, time is divided into slots, and multiple

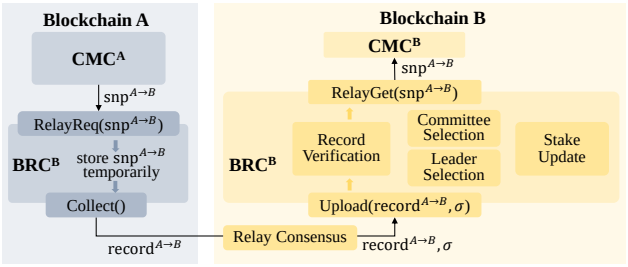


Fig. 5: Workflow of relaying *channel_snp*.

slots make up an epoch. Algorithm 1 shows the relay consensus process of relayers$^{A\to B}$ relaying channel snapshots from blockchain A to blockchain B. Explanations are as follows.

- *Committee selection (line 2).* At the beginning of each epoch j, all relayers are randomly divided into different groups. The group with the highest sum of stakes will then become the committee for epoch j, denoted as C_j. This process is executed by the CommitteeSelection function in BRC^B.
- *Leader selection (line 4).* For each slot t in epoch j, a leader l_t is randomly selected from the committee C_j in proportion to their stakes. For a relayer r_i with stake s_i, the probability of being selected as the leader is $p_i = \frac{s_i}{\sum_{i=1}^{n} s_i}$. We use a guaranteed-output delivery coin tossing protocol to implement the leader selection process. Let $\hat{p}_i = s_i / \Sigma_{k=i}^{n} s_k$. All relayers sequentially flip a $\hat{p}$-biased coin in the order of r_1, ..., r_n. The first relayer who obtains a result of 1 is selected as the leader, and the process terminates. It's worth noting that $\hat{p}_n = 1$, ensuring the selection of a unique leader. This process is executed by the LeaderSelection function in BRC^B.
- *Record consensus (line 5-18).* The leader l_t invokes the Collect interface in BRC^A to collect all the snapshots generated by CMC^A that should be relayed to CMC^B. For clarity, we denote the results as $\mathsf{record}_t^{A\to B}$. Each remaining relayer in the committee also invoke the Collect interface to acquire the snapshots, thus verifying whether l_i includes all snapshots in the generated record. If the record is valid, the relayer signs it and sending the signature to l_t. Subsequently, l_t can generate an aggregated signature σ when receiving signatures from more than $\frac{2}{3}$ of the relayers. Next, l_t invokes the Upload interface to submit $(\mathsf{record}_t^{A\to B}, \sigma)$ to BRC^B. BRC^B can verify the validity of the submitted record by checking whether σ is generated by the current committee.

2) Stake and Incentive: The primary goal of the incentive mechanism is to determine how to compute stake leveraged by the PoS consensus. As discussed in Section IV-D, the incentive framework should satisfy three principles: effectiveness, sustainability, and fairness. Among them, fairness can be easily guaranteed by leveraging smart contracts to execute the incentive mechanism. Additionally, to achieve effectiveness and sustainability, the incentive framework utilizes three mechanisms: reputation, deposit, and stake consumption.

- *Reputation*. Reputation reflects a relayer's behavior. Upon registration, each relayer is assigned an initial reputation score (r_0). Then, it increases by δ_r^+ when the relayer generates or votes for a correct record and decreases by δ_r^- if the record or vote is incorrect. Additionally, for a lazy relayer who does not participate in the consensus process, its reputation remains unchanged.
- *Deposit*. Relayers are required to deposit a fixed amount of assets, denoted by d_0, to the DPC when they register. If a relayer is selected as the leader to generate a record, it can receive rewards, denoted by δ_d. The rewards are added to its deposit and temporarily stored in the DPC. Subsequently, the relayer can withdraw its rewards. Relayers with different levels of cautiousness may choose to withdraw rewards at different frequencies, and the incentive goal is to encourage relayers to leave more assets on the DPC. This is important to guarantee asset availability when settling channels.
- *Stake consumption*. To achieve sustainability, the trend of stake variation should be relatively stable. In addition to using a logarithmic function to control the rate of change, we employ a stake consumption mechanism. This mechanism requires the leader to consume some stakes, i.e., to decrease the stake value, to generate the record and earn the reward. With this approach, we can ensure that the stake value fluctuates around the initial value, thus achieving sustainability.

Algorithm 1: Relay consensus of relayers$^{A\to B}$

```
1  for each epoch j do
2      C_j ← committee selection;
3      for each slot t do
4          l_t ← leader selection;
           // record consensus
5          record_t^{A→B} ← BRC^A.Collect();
6          σ_l ← Sign(prk_{l_t}, record_t);
7          σ̂.append(σ_l);
8          for each relayer r_i in C_j do
9              if verify (record_t, σ_l) success then
10                 σ_i ← Sign(prk_{r_i}, record_t);
11                 σ̂.append(σ_i);
12             end
13         end
14         if |σ̂| > (2/3)n then
15             σ ← aggregate signatures in σ̂;
16             BRC^B.Upload(record_t^{A→B}, asig);
17         end
18     end
19 end
```

Considering the above designs, the equation to compute the stake is defined as

$$s = \log(1 + \alpha d + \beta r - \eta l). \tag{1}$$

Here, d and r respectively represent the value of deposit and reputation, l is the time the relayer has acted as the leader. Notably, d, r, and l should be normalized as they may have different dimensions. Additionally, the three parameters, including α, β, η, are used to achieve sustainability. We quantify

sustainability by the difference between the initial stake and the average value during long-term operation. Specifically, let N denote the total number of relayers in the system, T denote the number of epochs considered as long-term, and s_{ij} denote the stake of relayer i after epoch j. The difference value (δ_s) can then be calculated as:

$$\delta_s = \mathrm{Avg}_{i\in[1,N]}(\mathrm{Avg}_{j\in[1,T]}(s_{ij})) - \mathrm{Avg}_{i\in[1,N]}(s_{i0}). \quad (2)$$

The closer δ_s is to 0, the stronger the sustainability is guaranteed. On this basis, the specific value of the parameters can be determined based on heuristic algorithms, and further discussion on specific results can be found in Section VI-B.

D. Extension to Contractless Blockchains

The proposed CrossChannel protocol is implemented using smart contracts. However, some blockchains, such as Bitcoin, only support simple scripts, precluding deployment of CrossChannel. Nevertheless, enabling complex smart contracts on such contractless blockchains remains an active research area, with emerging solutions facilitating CrossChannel deployment.

Several solutions exist, including side-chain execution, enclave execution, and off-chain execution. For instance, Rootstock (RSK) [34] establishes a sidechain that enables smart contracts for Bitcoin, allowing deploying the CrossChannel protocol on the sidechain. Enclave execution solutions, including Fastkiten [35] and Bitcointracts [36], leverage the trusted execution environment (TEE) technology to execute contracts, supporting arbitrary complex contracts. Off-chain execution, exemplified by Arbitrum [37] and BitML [38], allows an executor group to execute contracts off-chain, and adopt the multi-authentication mechanism to enable blockchain nodes to validate the off-chain execution results. That is, all executors collectively sign the result, and other blockchain nodes consider a result with enough signatures as valid.

While employing different techniques, all of the aforementioned solutions enable the execution of complex contracts on existing contractless blockchains, thus facilitating the extension of CrossChannel to contractless blockchains.

VI. Security Analysis

Our main idea is to realize the security of one-chain payment channels in cross-chain scenarios. Since the protocol adopts chain relay to realize security of cross-chain payment channel, we analyze CrossChannel's security from two aspects. First, the channel is secure if the chain relay is secure (VI-A). Second, the chain relay is secure (VI-B). In addition to security of the channel, in cross-chain scenarios, the exchange rate between blockchains also effects the correctness of payments, and we finally discuss the issue (VI-C).

A. Security of the Channel Management Protocol

In CrossChannel, each channel is maintained by two CMCs, i.e., CMC^A and CMC^B, and our primary goal is to ensure the consistency between them (Theorem 1). With consistency, the cross-chain channel is then equivalent to an one-chain channel, and the security can be proved similarly (Theorem 2).

Theorem 1. ***Consistency.*** *The two CMCs maintain the same* Channel *for the same cross-chain channel. If one CMC (e.g., CMC^A) makes any change on* Channel*, the relative CMC (CMC^B) will also change to the same value or let CMC^A revert back to the previous value.*

Proof. As depicted in Fig. 3, a Channel consists of the channel_info and channel_snp formed by cid and state. All items of channel_info are immutable, and are compressed in cid. Therefore, if two Channel have the same cid, they must have the same channel_info. Thus, the consistency of channel_info is easily guaranteed, and our main purpose is to analyze the consistency of state.

There are four potential channel states: init, activated, preclosed[bp], and closed[bp]. Additionally, there exists a special state wherein the channel does not exist. For clarity, we use i, a, p^{bp}, and c^{bp} to represent the four states respectively, and employ _ to indicate the inexistence of the channel. Notably, if two CMCs have the same state p or c but with deferring bp parameters, they are inconsistent. To prove the consistency, we utilize the state pair notation to represent the channel state that maintained separately by each CMC, taking the form of $(\mathsf{state}^A, \mathsf{state}^B)$.

Fig. 6 demonstrates the state transition diagram, where each node represents a state pair, and the edge shows users' operation and required time. Initially, both CMCs are _. When a user, suppose U_1^A, invokes $\mathrm{CreateRequest}$, the state pair changes to $(\mathsf{i}, \_)$. If U_2^B invokes $\mathrm{CreateRequest}$ in CMC^B, the relayed snapshot snp_2^B will have the same cid with that in CMC^A. As a result, CMC^A changes state to a. Otherwise, it reverts to _. Since the process is symmetric between CMC^A and CMC^B, they will both reach a or revert to _. When the state pair reach (a, a), the two users can conduct off-chain payments through the activated channel.

When a user decides to close the channel, recall the four steps of channel closing (Section V-B3), the state transitions are divided to four steps accordingly. First, when a user, e.g., U_1^A, invokes $\mathrm{CloseRequest}$ in CMC^A, the state pair changes to $(\mathsf{p}^{\mathsf{bp}}, \mathsf{a})$, and the snapshot will be relayed to CMC^B. Then, CMC^B takes corresponding actions and changes state^B, leading to three potential state^B, including a, c^{bp}, and $\mathsf{p}^{\mathsf{bp}'}$. Subsequently, state^B is relayed to CMC^A, and CMC^A takes actions accordingly. For state^B equals a or c^{bp}, CMC^A reverts to a or close the channel with bp, and the closure process is finished. Otherwise, if state^B is $\mathsf{p}^{\mathsf{bp}'}$, the fourth step is required to help CMC^B determine which balance proof should be used to close the channel. In conclusion, the channel closure process has three results: both revert to a, both close the channel with bp, or both close the channel with bp′. Thus, consistency is guaranteed. □

Theorem 2. ***Correctness.*** *The channel only closes with the correct balance proof which satisfies three requirements: ① it has two signatures from the two users; ② its total balance equals to the initial locked assets when creating the channel; ③ it is the last balance proof of the channel.*

Proof. Requirement ① is easily satisfied because signatures are directly verified by the blockchains. Requirement ② and

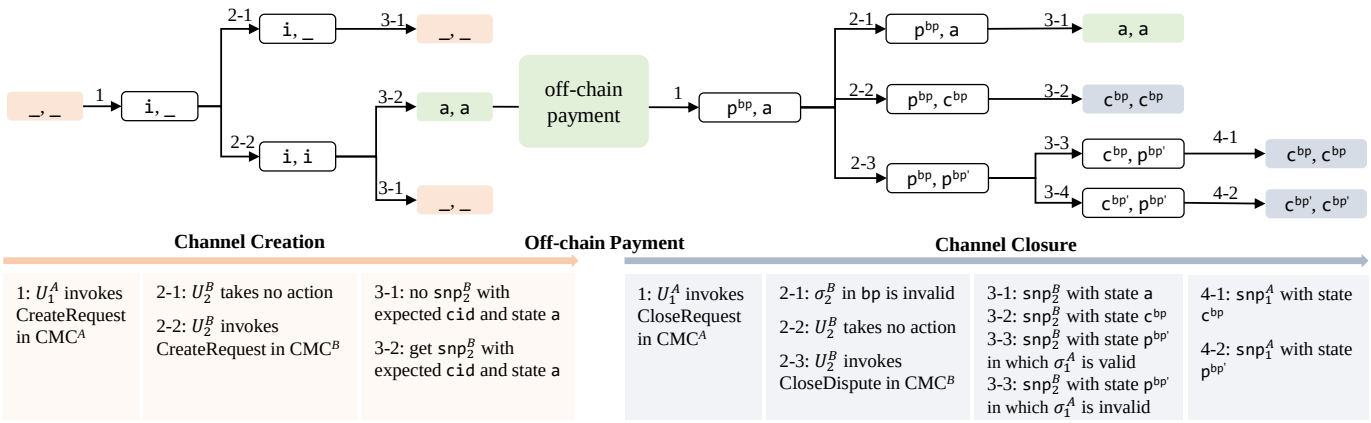


Fig. 6: Transition diagram of the state pair.

③ are corresponding to the colluded attack and mutual attack of users (see Section IV-C), respectively. If two users collude to generate a balance proof in which the total balance exceeds their locked assets, CMCs can detect the issue since they confirmed the initial assets during channel creation. Besides, if one user submitted an outdated balance proof to close the channel, the other one can dispute from his/her CMC, and Theorem 1 proves that as long as the dispute is valid, the channel will close with it. Thus, correctness is guaranteed. □

B. Security of the Chain Relay protocol

The primary security goal of the chain relay protocol is to ensure the correct relay of all channel_snp. We prove the security from two aspects. First is the bootstrapping security, which guarantees the secure deployment and bootstrap of the chain relay protocol on a blockchain from the beginning. Second is the long-term security, which refers to the sustained healthy operation facilitated by the incentive mechanism.

Theorem 3. *Bootstrapping security. The chain relay protocol can be securely deployed and bootstrapped on a blockchain when $\rho < \frac{1}{2}$, with the assistance of audit.*

Proof. At the very beginning, a group of relayers initiate the bridge contract (BRC) and lock their deposits. To bootstrap the chain relay protocol, an arbitrary relayer invokes the StakeUpdate function to assign initial stakes and invokes the CommitteeSelection function to generate the first committee. The selection of the first committee is purely random because all relayers are assigned the same stake value initially.

As we discuss in Section IV-D, ρ relayers may be compromised by users and try to submit forged records. If there are more than $\frac{2}{3}$ malicious relayers in the committee, they can manipulate BRC into accepting a forged record. We refer to such a committee as a failed committee. Assuming there are N relayers in total and the committee size is n, the probability of selecting a failed committee, denoted as $P_{failComm}$, is

$$P_{failComm} = \left[\sum_{i=0}^{n-\lfloor \frac{2n}{3} \rfloor} \binom{\rho N}{\lfloor \frac{2n}{3} \rfloor + i} \binom{(1-\rho)N}{\lfloor \frac{n}{3} \rfloor - i} \right] \Big/ \binom{N}{n}.$$

In addition, if the proportion of malicious relayers in the committee is between $\frac{1}{3}$ and $\frac{2}{3}$, no record will be accepted by BRC since they cannot receive enough votes, whether correct or error. This will lead to an empty slot, i.e., no valid record is submitted to BRC. In such a case, BRC can launch a new committee selection, and the snp that failed to be relayed will be contained in the following relay record.

According to the general security assumption adopted by Byzantine fault tolerance consensus algorithms, we focus on the results when $\rho = \frac{1}{3}$. TABLE I shows the values of $P_{failComm}$ when N equals 500, 1000, and 2000, and n equals $0.02N$, $0.05N$, and $0.1N$, respectively. The results demonstrate that $P_{failComm}$ is a relatively small probability but should not be considered negligible, especially when N and n are small. This implies that there is a chance of selecting a failed committee. However, this will not result in security breaches because the relay process is designed to be auditable. Specifically, relayers obtain the relay record from the Collect interface, ensuring that all relayers can access the record. Thus, if a failed committee submits an incorrect record to BRC, other relayers can detect the error. They can then submit evidence consisting of the correct record and an aggregated signature generated by at least $(1-\rho)N$ relayers. When BRC receives a valid evidence, it replaces the incorrect record and reduces the reputation of relayers who signed the incorrect record. Therefore, when $\rho = \frac{1}{3}$, bootstrapping security can be guaranteed with the assistance of audit. In fact, as long as $\rho < \frac{1}{2}$, the audit process can perform well. To conclude, bootstrapping security can be guaranteed with the assistance of audit when $\rho < \frac{1}{2}$. □

TABLE I: VALUES OF $P_{failComm}$ WHEN $\rho = \frac{1}{3}$

$P_{failComm}$	$n = 0.02N$	$n = 0.05N$	$n = 0.1N$
$N = 500$	0.02	2.6×10^{-4}	9.6×10^{-8}
$N = 1000$	6.9×10^{-4}	2.2×10^{-7}	2.1×10^{-13}
$N = 2000$	6.8×10^{-6}	9.8×10^{-13}	2.6×10^{-25}

Theorem 4. *Long-term security. The incentive framework satisfies effectiveness, sustainability, and fairness, ensuring that the relay protocol operates securely for the long-term.*

Proof. As explained in Section IV-D, an incentive mechanism should adhere to several principles including effectiveness,

sustainability, and fairness. In the proposed scheme, fairness is intuitively guaranteed since the incentive mechanism is executed by smart contracts, which are safe from manipulation. Effectiveness and sustainability are guaranteed by Eq. (1) and Eq. (2). In Eq. (1), the deposit d and the reputation r are positively correlated parameters. Thus, relayers with higher deposits and reputations can obtain larger stakes, increasing their chances of acting as the leader and earning rewards. On the contrary, malicious relayers who sign incorrect records will lose their reputations. When a relayer loses all its reputation value, its deposit will be confiscated, creating a deterrent that prevents relayers from engaging in malicious behavior. Thus, effectiveness is guaranteed. Furthermore, once a relayer acts as a leader, its stake will be reduced, preventing the relayer's stake from increasing indefinitely. Assisted by Eq. (2), the difference in stake values among relayers after long-term operation is limited to a small value, ensuring that all honest relayers have the same expected state value and guaranteeing sustainability. □

To demonstrate long-term security, we simulate the long-term operation of the relay process. To run the algorithm, we first set the system parameters as follows: the total number of relayers N=1000, the committee size n=50, the number of slots in an epoch t=100, and the long-term operation represented by T=10000 epochs. Second, for normalization, we set δ_d=1 and $\delta_r^+=\frac{1}{n}$, ensuring that the increasing rate of deposit and reputation is equal to that of l. Third, the initial deposit and reputation are fixed at d_0=r_0=100, and the withdrawal policy is set such that each relayer will withdraw all rewards when they accumulate $\frac{1}{2}d_0$ rewards.

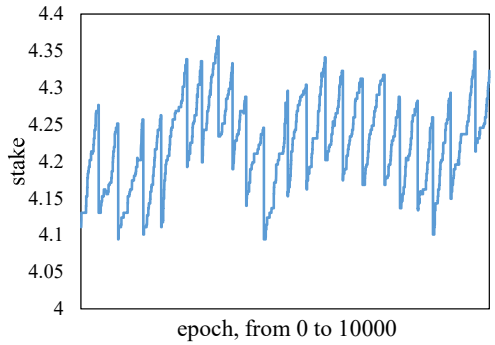
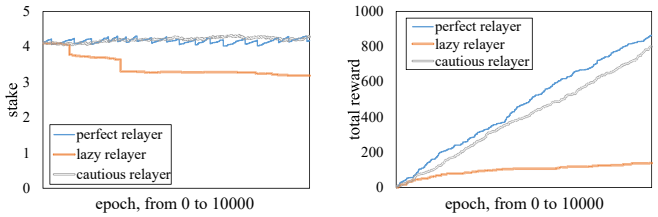


Fig. 7: Variation trend of stake during 10000 epochs.

Based on the above settings, we determine the values of parameters in Eq. (1) using a Particle Swarm Optimization (PSO) algorithm, with the objective of minimizing δ_s in Eq. (2). The PSO algorithm outputs $(0.2, 0.4, 0.4)$ as a possible solution of (α, β, η), with $\delta_s = 0.008$. The corresponding stake variation trend is shown in Fig. 7, showing that the stake fluctuates between 4.1 and 4.4. This implies that the relayers' stakes are basically stable, ensuring that all honest relayers, whether new or old, have a similar chance to acquire rewards, thus proving sustainability. It is easy to notice that the stake will experience periodic drops, which are caused by relayers withdrawing their rewards.



(a) variation trend of stake (b) variation trend of total reward

Fig. 8: Variation trend of relayer's stake and total reward.

For effectiveness, we analyze the stake variation trend for relayers who act differently. First, we analyze how the incentive mechanism can prevent lazy relayers and cautious relayers. We assume that a lazy relayer, when not selected as a leader, has an 80% probability of not participating in the consensus process. A cautious relayer will withdraw its rewards every $\frac{1}{10}d_0$ rewards it receives. In contrast, a perfect relayer will fully participate in the consensus process and only withdraw its rewards after receiving $\frac{1}{2}d_0$ rewards. Based on these assumptions, we evaluate the variation trend of stake and total reward for the three types of relayers. The results are shown in Fig. 8. From Fig. 8(a), we can see that if a relayer is lazy, its stake will gradually decrease and stabilize at a lower value, while the stakes of perfect relayers and cautious relayers are relatively close. However, Fig. 8(b) shows that cautious relayers earn fewer rewards compared to perfect relayers, implying that the incentive mechanism encourages relayers to leave more assets in DPC. In addition, lazy relayers earn significantly fewer rewards. These results demonstrate that the proposed incentive framework satisfies effectiveness by preventing lazy relayers and cautious relayers.

C. Discussion of the Exchange Rate

The exchange rate is a crucial factor in cross-chain transactions. There are several methods to obtain a publicly acknowledged exchange rate, as mentioned in [39] and [40]. Therefore, we assume the existence of a publicly acknowledged exchange rate and require relayers to record it on-chain. For example, they can periodically submit the current exchange rate in BRC. To ensure correctness, they can leverage the relay consensus to submit the exchange rate.

Malicious users may deliberately use the wrong exchange rate to benefit themselves. For example, if $\gamma(A, B) = 2$ but the user sets $\gamma(A, B) = 3$, they will gain an additional coin^B after paying one coin^A through the cross-channel. However, this is not feasible in the proposed scheme because the exchange rate is publicly recorded on-chain. CMC will verify its correctness when creating the channel and will check whether the change of balance complies with this rate when closing the channel.

This solution eliminates the possible malicious acts of users but also imposes a limitation on flexibility. All transactions within a channel must adhere to the same exchange rate. If the exchange rate undergoes significant changes, the channel users may incur a loss. Nonetheless, we still believe that establishing a cross-channel is more efficient and cost-effective

for users who need to conduct a large number of micro cross-chain transactions, especially when the exchange rate remains relatively stable or changes infrequently.

VII. Performance Analysis

We develop a prototypical implementation between two Ethereum test networks. For comparison, we also implement the HTLC-based atomic swap scheme. To show the feasibility of CrossChannel, i.e., we compare the two schemes in terms of latency and expense. Results powerfully demonstrate that CrossChannel makes excellent strides in the scalability and affordability of cross-chain transactions.

A. Latency and Execution Cost

1) Minimum Latency: First, we analyze the transaction process of HTLC swap and CrossChannel to compute their minimum latency. We use block cost, which represents the number of blocks needed to complete a transaction, to denote the minimum latency. This is because the block generation speed is relatively stable and considered a primary unit of time. As shown in Fig. 9, each CCTx through HTLC swap requires a total of four blocks. On the other hand, establishing a cross-channel requires six blocks, with three on each blockchain if the relay frequency equals the block generation frequency. Since the establishment processes on the two chains are concurrent, the overall time consumption for channel establishment is thus three blocks' generation time. Closing a cross-channel requires seven blocks.

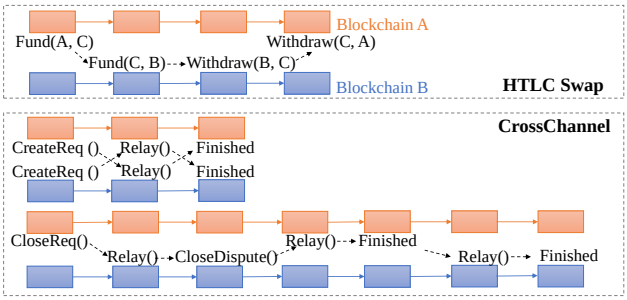


Fig. 9: Block cost of HTLC and CrossChannel.

It may seem that CrossChannel requires more than twice as many blocks as HTLC swap. However, the HTLC-based CCTx is independent, which means that each CCTx requires four blocks separately. On the other hand, once a cross-channel is established, the two users can freely conduct CCTxs through the channel and off-chain, without incurring any additional block cost. Therefore, as the number of CCTxs in a channel increases, the average block cost per CCTx will continuously decrease, as shown in Fig. 10. When considering the average block cost, CrossChannel acquires better performance.

2) Execution Cost: The execution cost refers to the overhead of invoking the related smart contracts and completing specific functions. Ethereum provides a clear indicator for execution cost, which is gas. The overhead of each operation in Ethereum smart contracts is evaluated by a certain gas value. After successfully invoking a contract, the gas consumption is recorded in the related transaction. Therefore, we evaluate the gas consumption of both HTLC-swap and CrossChannel to compare their execution cost.

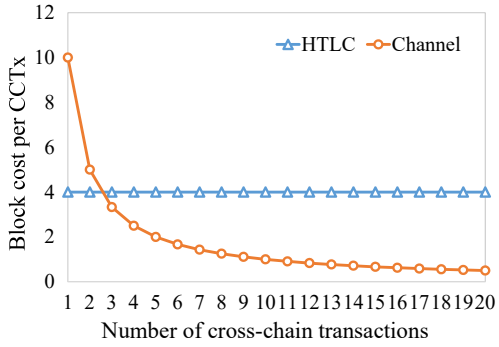


Fig. 10: Block cost of HTLC swap and CrossChannel.

TABLE II: Main step's execution cost

Step	HTLC		CrossChannel		
	Fund	Withdraw	CreateRequest	CloseRequest	CloseDispute
Gas	152,874	87,172	642,912	172,357	93,537
Ether	0.0084	0.0048	0.0371	0.0130	0.0069

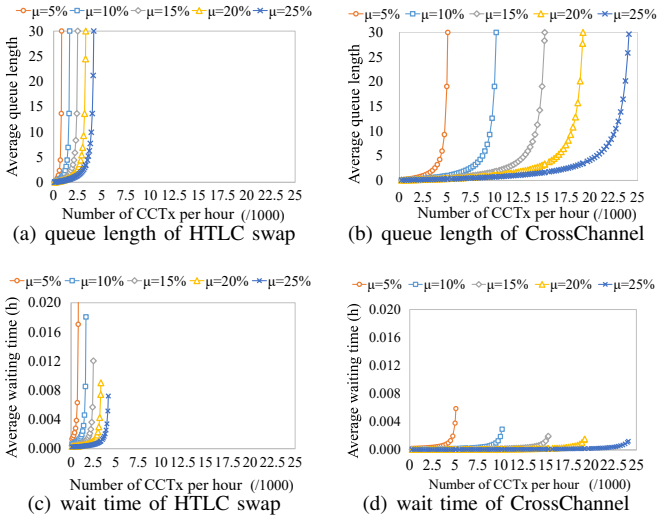
TABLE III: Participants' execution cost

Item	HTLC				CrossChannel		
	sender	recipient	agent	total	user1	user2	total
Gas	152,874	87,172	240,046	480,092	815,269	736,449	1,551,718
Ether	0.0084	0.0048	0.0132	0.0264	0.0448	0.0405	0.0853

TABLE II shows the cost of key functions of HTLC and CrossChannel. As depicted in Fig. 9, HTLC swap consists of Fund and Withdraw, CrossChannel consists of CreateRequest, CloseRequest, and CloseDispute. Based on these results, we can acquire the cost of each participant and the total cost. For HTLC swap, there are three participants in a transaction, i.e., sender, recipient, and agent. First, the sender invokes Fund in the HTLC contract provided by his/her blockchain, and the agent then invokes Fund in the recipient's blockchain. Subsequently, the recipient invokes Withdraw in the local blockchain, and the agent finally invokes Withdraw in the sender's blockchain to finish the swap. For CrossChannel, the two users execute the same process to create the channel. When closing the channel, one user acts as the closing requester and the other acts as the disputer, and we denote them as user1 and user2, respectively. As we can see in TABLE III, the total cost of CrossChannel is about three times as much as that of HTLC swap.

3) Systematical Performance: Based on the above experiments results, we systematically compare the performance of the proposed CrossChannel to that of HTLC atomic swaps on different transaction frequencies using a $(M, M, 1)$ queuing model, results are shown in Fig. 11. We regard the two schemes as CCTx service providers and use an hour as the time unit. Therefore, the average arrival rate λ is the number

of CCTx per hour, and the average service duration $1/\mu$ is computed as follows.



(a) queue length of HTLC swap (b) queue length of CrossChannel

(c) wait time of HTLC swap (d) wait time of CrossChannel

Fig. 11: Average queue length and wait time of HTLC swap and CrossChannel.

For the HTLC swap, a CCTx consists of two parallel Fund-Withdraw pairs, one on each blockchain. The total gas consumption of each pair is $240,046$, while the current gas limitation of an Ethereum block is $15,000,000$. Thus, the transactions per block (TPB) of HTLC-swap CCTx is 62. Considering that the average block generation time is 13 seconds, the transactions per hour (TPH) of HTLC-swap CCTx is then $17,169$. Similarly, the CrossChannel consists of channel creation and channel closing, and the closing includes close request and close dispute, and we use the average of them as the closing overhead. Thus the total gas consumption of an entire CCTx is $642,912 + avg(172,357,\ 93,537) = 775,859$, then TPB and TPH are 19 and $5,261$ respectively. However, in a practical blockchain, it is impossible that CCTx occupies the entire block. Besides, for CrossChannel CCTx, we should consider the proportion of off-chain transactions because only creating or closing a channel will incur system overhead. Hence, we assume that the proportion of CCTx to all transactions is p, the proportion of off-chain transactions is $1-\epsilon$, and the average service duration $1/\mu$ is $17,169p$ and $5,261p$ separately for HTLC swap and CrossChannel. TABLE IV shows a summary of the above computing process.

TABLE IV: SERVICE DURATION COMPUTATION

Params	Steps	Gas	TPB	TPH $(1/\mu)$
HTLC	fund & withdraw	$240,046$	62	17169
Channel	creation & closing	$775,859$	$19/\epsilon$	$5261/\epsilon$

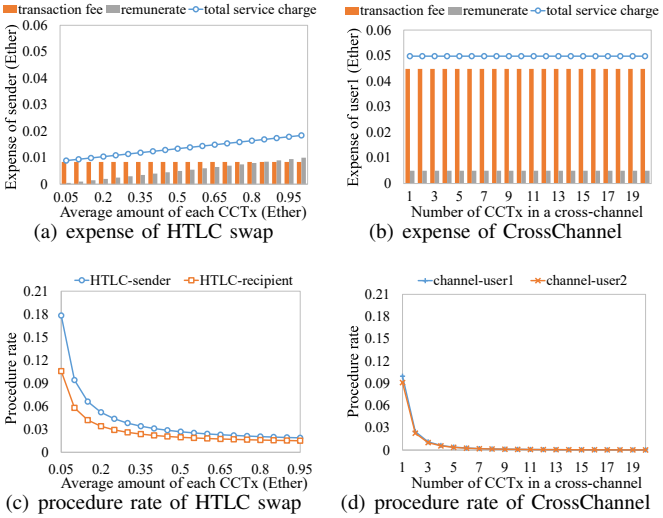
TABLE V: PERFORMANCE COMPARISON

Params and Indicators	Length Limitation	p	ϵ	Max Capacity (CCTx per hour)	Max Waiting Time (h)
HTLC	30	25%	/	4155	0.007
Channel	30	25%	5%	24000	0.001

Based on the above results, we evaluate the systematical performance of the two schemes. After conducting some testing experiments, we set the range of average arrival rate λ as $[0, 25,000]$ and $p \in \{5\%, ..., 25\%\}$. Additionally, ϵ represents the average number of CCTx in each cross-channel. We assume the number is 20, thus $\epsilon = 5\%$. We choose the average queue length l and the average waiting time w as the performance indexes, and results are shown as Fig. 11. For clarity, we set the same coordinate intervals for HTLC-swap and CrossChannel. From Fig. 11(a) and 11(b), we can observe that when $\mu = 25\%$ and the queuing length upper bound is set as 30, HTLC-swap can handle up to $5,000$ CCTx per hour, while CrossChannel approaches $25,000$. Moreover, Fig. 11(c) and 11(d) demonstrate that the average waiting time for each CCTx of CrossChannel is significantly lower than that of HTLC swap. We provide a summary in TABLE V for reference. In conclusion, when each channel consists of 20 transactions, CrossChannel has a fivefold transaction capacity compared to the HTLC-swap. It should be noted that the number of transactions in each channel may be much higher than 20 in practice, leading to even better performance for CrossChannel.

B. Transaction Expense

In addition to reducing the transaction latency, another important advantage of establishing an off-chain payment channel is reduction of the expenses. To evaluate how effective CrossChannel could be in terms of expense, we compare the total transaction expense of HTLC swap and CrossChannel, including transaction fee and brokerage for the agent or relayer. The results are shown as Fig. 12.



(a) expense of HTLC swap (b) expense of CrossChannel

(c) procedure rate of HTLC swap (d) procedure rate of CrossChannel

Fig. 12: Expense and procedure rate of HTLC swap and CrossChannel.

The transaction fee is the cost to invoke the related smart contracts and is evaluated in gas in Ethereum. For clarity, we convert gas to Ether according to the current gas fee, i.e., 1 gas = 55 GWei. The brokerage is the fee paid to the intermediary in HTLC swap and to relayers in CrossChannel,

and we set the brokerage rage as 1%, that is, for each CCTx of 1 Ether, a brokerage of 0.01 Ether is needed. Based on the above settings, we evaluate the average expense of each CCTX and the procedure rate. Fig. 12(a) and 12(b) are about the expense. For HTLC swap, the transaction fee is constant, but the brokerage changes with the amount of a single CCTx. Therefore, we record the expense of each CCTx with different amounts for a single CCTx. For CrossChannel, the transaction fee is also constant. Still, the brokerage is difficult to directly compute through the amount of a single CCTx, because the inner-channel transactions do not cause blockchain overhead. Therefore, we use the average on-chain transaction amount to compute the brokerage and set it as 0.05 Ether by roughly counting Ethereum transaction amounts. Thus, the brokerage of CrossChannel is also constant and does not change with the number of transactions in a channel. Based on the expense, we compute the expense rate and the results are shown in Fig. 12(c) and Fig. 12(d). It is clear that, although the expense of CrossChannel seems more than that of HTLC swap, the expense rate of each transaction of CrossChannel is much lower. Even when the average amount of a single CCTx is up to 1 Ether, which is a large amount in the actual Ethereum, the expense rate of HTLC swap is still much higher than that of CrossChannel. To conclude, CrossChannel performs much better than HTLC swap in transaction expense.

VIII. Conclusion

In this paper, we proposed CrossChannel, which enables two non-interoperable users, where each individual only has accounts on one chain, to establish a cross-and-off-chain micropayment channel for fast and cost-effective cross-chain transactions. Specifically, we analyzed the security and availability issues of extending the existing one-chain payment channel protocol to cross-chain scenarios and leveraged the chain relay protocol to solve them. Additionally, to ensure the efficiency and security of chain relay, we proposed a channel snapshot mechanism that packages all required information in a concise snapshot, acting as the relay content. Furthermore, we designed a proof-of-stake (PoS) consensus algorithm supported by an incentive mechanism, which is adopted among relayers to ensure the correct relay of the snapshot.

To evaluate the security and effectiveness of the proposed CrossChannel, we conducted comprehensive security and performance analyses. In terms of security, we demonstrated that CrossChannel meets the security requirements as an off-chain payment channel and provided proof of both bootstrapping and long-term security of the chain relay protocol. In terms of performance, we implemented CrossChannel as a prototype between two Ethereum testnets and also implemented the HTLC-swap scheme for comparison. We evaluated the latency and transaction fee of both schemes, and the results show that CrossChannel is significantly more scalable and affordable than HTLC swap, providing strong evidence for the effectiveness of our proposed CrossChannel.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371 and No. U19B2023, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchze, A. Kiayias, and W. J. Knottenbelt, "SoK: Communication across distributed ledgers," in *Proceedings of the 2021 International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2021, pp. 3–36.

[2] C. Dougherty and G. Huang, "Mt. gox seeks bankruptcy after 480 million dollars bitcoin loss," *Bloomberg News*, 2014, accessed: Jun., 2022. [Online]. Available: https://www.bloomberg.com/news/articles/2014-02-28/mt-gox-exchange-files-for-bankruptcy

[3] C. Baldwin, "Bitcoin worth 72 million Dollars stolen from bitfinex exchange in hong kong," *Reuters*, 2016, accessed: Jun., 2022. [Online]. Available: https://www.reuters.com/article/us-bitfinex-hacked-hongkong-idUSKCN10E0KP

[4] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM symposium on principles of distributed computing (PODC)*. ACM, 2018, pp. 245–254.

[5] M. Herlihy, B. Liskov, and L. Shrira, "Cross-chain deals and adversarial commerce," *The VLDB journal*, pp. 1–19, 2021.

[6] Y. Hirai, "Blockchains as kripke models: An analysis of atomic cross-chain swap," in *Proceedings of the 2018 International Symposium on Leveraging Applications of Formal Methods (ISOLA)*. Springer, 2018, pp. 389–404.

[7] S. Thomas and E. Schwartz, "A protocol for interledger payments," 2015, accessed: Jun., 2022. [Online]. Available: https://interledger.org/interledger.pdf

[8] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 1521–1538.

[9] P. Gazi, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 139–156.

[10] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *Proceedings of the 2019 International Conference on Financial Cryptography and Data Securit (FC)*. Springer, 2019, pp. 21–34.

[11] F. Gai, J. Niu, C. Grajales, M. M. Jalalzai, and C. Feng, "A secure consensus protocol for sidechains," https://arxiv.org/abs/1906.06490, 2019, arXiv preprint.

[12] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant paymetns," 2018, Lightning Network white paper, accessed: Jun., 2022. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf

[13] "Alt chains and atomic transfers," https://bitcointalk.org/index.php?topic=193281.msg2003765, accessed: Jun., 2022.

[14] S. Imoto, Y. Sudo, H. Kakugawa, and T. Masuzawa, "Atomic cross-chain swaps with improved space and local time complexity," in *Proceedings of the 2019 International Symposium on Stabilizing, Safety, and Security of Distributed Systems (SSS)*. Springer, 2019, pp. 194–208.

[15] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability," in *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. NDSS, 2019.

[16] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "XCLAIM: Trustless, interoperable, cryptocurrency-backed assets," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019.

[17] A. Garoffolo, R. Viglione, and R. Oliynykov, "Zendoo: A zk-SNARK verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains," in *Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 1257–1262.

[18] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 473–489.

[19] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 455–471.

[20] E. Heilman, L. AlShenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS)*. NDSS, 2017.

[21] S. Dziembowski, S. Faust, and K. Hostakova, "General state channel networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 949–966.

[22] V. Sivaraman, B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2020, pp. 777–796.

[23] E. Heilman, S. Lipmann, and S. Goldberg, "The arwen trading protocols," in *Proceedings of the 2020 International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2020, pp. 156–173.

[24] J. A. Mathus-Garza, "The lightning network cross-chain exchange: a decentralized approach for peer to peer exchange across blockchain," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[25] "Raiden network," accessed: Jun., 2022.

[26] "BTC Relay," https://github.com/ethereum/btcrelay, accessed: Jun., 2022.

[27] M. Westerkamp and J. Eberhardt, "zkRelay: Facilitating sidechains using zkSNARK-based chain-relays," in *Proceedings of the 2020 IEEE European Symposium on Security and Privacy Workshops (SPW)*. IEEE, 2020, pp. 378–386.

[28] A. Gervais, G. O. Karame, K. W´ust, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (CCS)*. ACM, 2016, pp. 3–16.

[29] A. Kiayias, A. Russel, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proceedings of the 2017 Annual International Cryptology Conference (ASIACRYPT)*. Springer, 2017, pp. 357–388.

[30] G. Xu, Y. Liu, and P. W. Khan, "Improvement of the DPoS consensus mechanism in blockchain based on vague sets," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4252–4259, 2019.

[31] X. Feng, J. Ma, Y. Miao, X. Liu, and K.-k. R. Choo, "Social characteristic-based propagation-efficient PBFT protocol to broadcast in unstructured overlay networks," *IEEE Transactions on Dependable and Secure Computing*, 2021, DOI:10.1109/TDSC.2021.3104465.

[32] V. Buterin, "Ethereum proof-of-stake consensus specifications," 2021, accessed: Jun., 2022. [Online]. Available: https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/beacon-chain.md

[33] R. Han, Z. Yan, X. Liang, and L. T. Yang, "How can incentive mechanisms and blockchain benefit with each other?" *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38, 2022.

[34] "Rootstock: Smart contracts secured by bitcoin," https://rootstock.io/, accessed: Nov., 2023.

[35] P. Das, L. Eckey, T. Frassetto, D. Gens, K. Hostáková, P. Jauernig, S. Faust, and A.-R. Sadeghi, "FastKitten: Practical smart contracts on bitcoin," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security)*. USENIX Association, 2019, pp. 801–818.

[36] K. Wüst, L. Diana, G. Kostiainen, S. Matetic, and S. Capkun, "Bitcontracts: Supporting smart contracts in legacy blockchains," in *Proceedings of the 2021 Network and Distributed System Security Symposium (NDSS)*. NDSS, 2021, pp. 1–18.

[37] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security)*. USENIX Association, 2018, pp. 1353–1370.

[38] M. Bartoletti and R. Zunino, "BitML: A calculus for bitcoin smart contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 83–100.

[39] X. Li and C. A. Wang, "The technology and economic determinants of cryptocurrency exchange rates: The case of bitcoin," *Decision support systems*, vol. 95, pp. 49–60, 2017.

[40] T. Tarasova, O. Usatenko, A. Makurin, V. Ivanenko, and A. Cherchata, "Accounting and features of mathematical modeling of the system to forecast cryptocurrency exchange rate," *Accounting*, vol. 6, no. 8, pp. 357–364, 2020.

Xinyi Luo received her B.S. degree in Information Security from School of the Gifted Young, University of Science and Technology of China (USTC), in 2020. She is currently working toward the Ph.D. degree in Information Security with the School of Cyber Science and Technology, USTC. Her research interests include Blockchain, Network security and Applied Cryptography.

Kaiping Xue (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his doctor's degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Science and Technology, USTC. He is also the director of Network and Information Center, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. His work won best paper awards in IEEE MSN 2017 and IEEE HotICN 2019, the Best Paper Honorable Mention in ACM CCS 2022, the Best Paper Runner-Up Award in IEEE MASS 2018, and the best track paper in MSN 2020. He serves on the Editorial Board of several journals, including the IEEE Transactions on Dependable and Secure Computing (TDSC), the IEEE Transactions on Wireless Communications (TWC), and the IEEE Transactions on Network and Service Management (TNSM). He has also served as a (Lead) Guest Editor for many reputed journals/magazines, including IEEE Journal on Selected Areas in Communications (JSAC), IEEE Communications Magazine, and IEEE Network. He is an IET Fellow and an IEEE Senior Member. He is the corresponding author of this paper.

Qibin Sun (F'11) received the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1997. He is currently a professor in the School of Cyber Science and Technology, USTC. His research interests include multimedia security, network intelligence and security, and so on. He has published more than 120 papers in international journals and conferences. He is a fellow of IEEE.

Jun Lu received his bachelor's degree from southeast university in 1985 and his master's degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1988. Currently, he is a professor in the School of Cyber Science and Technology and the Department of EEIS, USTC. His research interests include theoretical research and system development in the field of integrated electronic information systems, network and information security. He is an Academician of the Chinese Academy of Engineering (CAE).