Lizz: Bitcoin-Compatible, Conditional and Flexible Multi-Party Payment Channel

Leiyang Wei, Zhicheng Xu, Yiqiao Song, and Hongbing Cheng
(Corresponding author: Hongbing Cheng)

School of Computer Science and Technology & Zhejiang University of Technology
Zhejiang University of Technology, Hangzhou, China
Email: chenghb@zjut.edu.cn

(Received Mar. 4, 2025; Revised and Accepted Nov. 26, 2025; First Online Apr. 6, 2026)

Abstract

The multi-party payment channel is a key solution for scaling blockchain. However, existing solutions face two issues: (1) *Lack of Bitcoin Compatibility:* They rely on smart contracts, unsupported by Bitcoin's scripting. Given Bitcoin's dominance, a compatible solution is crucial. (2) *Absence of Conditional Payments:* Current channels cannot enforce payments based on real-world conditions, limiting applications like supply chain finance. We propose Lizz, a Bitcoin-compatible multi-party conditional payment channel. Lizz uses a user-managed committee to track balances, verify transactions, and arbitrate disputes on Bitcoin. It employs threshold signatures to approve off-chain updates only when conditions are met, deterring fraud via penalty transactions. Theoretical analysis shows Lizz ensures security. Experiments in a 96-user scenario show Lizz cuts costs and reduces overhead by 77.14% compared to Bitcoin-compatible TPC schemes while achieving low latency and high scalability.

Keywords: Bitcoin; Conditional Payment; Multi-Party; Payment Channel

1 Introduction

Blockchain payment channels are widely regarded as a promising solution to scalability challenges [29]. While they enhance off-chain throughput, traditional payment channels primarily support transactions between two parties [27], limiting their applicability in multi-party payment scenarios. A prominent example is supply chain finance, where multiple entities—including suppliers, manufacturers, and retailers—engage in transactions often contingent on the delivery of physical goods [8]. In such cases, traditional payment channels necessitate direct connections between all involved parties, leading to increased costs and scalability constraints. Therefore, designing an efficient payment channel solution that supports transactions among multiple parties is necessary.

Existing approaches for multi-party payments fall into two categories. The first relies on intermediaries, with Payment Channel Networks (PCNs) [3, 6, 21, 22] as a key example. PCNs connect multiple two-party channels, facilitating indirect payments and minimizing the need for numerous direct links. However, they are limited to one-to-one payments and become inefficient in multi-party scenarios with frequent many-to-one or one-to-many transactions, as routing complexity and fees increase significantly. To address this, the second category extends PCNs or two-party channels to support multi-party payments. Studies [5, 12, 14] have improved PCNs for many-to-one transactions, while others [7, 28] introduce a leader or operator to manage updates, enhancing efficiency and flexibility. Despite these advancements, existing multi-party payment solutions still face two major issues:

- **Lack of Bitcoin Compatibility**: These solutions rely on smart contracts to coordinate channel operations, rendering them incompatible with Bitcoin's limited scripting capabilities [2, 13]. As of March 11, 2024, Bitcoin's market capitalization stands at $1,345B on Binance Exchange [4], significantly surpassing other cryptocurrencies. This highlights the critical need for Bitcoin-compatible multi-party payment solutions.

- **Absence of Conditional Payment Support**: These approaches do not support conditional payments, which are essential for cryptocurrency transactions tied to real-world events [19]. While existing conditional payment channel solutions [19] address this, they remain restricted to two-party channels and are incompatible with Bitcoin's non-Turing-complete scripting environment, limiting their general applicability.

In this paper, we propose Lizz, a Bitcoin-compatible multi-party payment channel protocol with conditionality and flexibility. The core idea of Lizz is to replace smart

contracts in traditional payment channels with a *user-participated committee*, enabling coordination and dispute arbitration on Bitcoin, which lacks Turing completeness. Specifically, the committee verifies payment conditions and enforces conditional payments. Transactions proceed only when conditions are met; otherwise, the committee has the authority to penalize malicious participants. Furthermore, Lizz introduces a proxy role that coordinates users joining or exiting the channel based on conditional payments.

Designing a Bitcoin-compatible multi-party conditional payment protocol is challenging due to Bitcoin's restricted scripting, lack of stateful execution, and absence of native support for off-chain multi-party coordination. Unlike Ethereum, Bitcoin's UTXO model lacks global state tracking, making it difficult to synchronize multi-party balances and prevent rollback attacks. Additionally, Bitcoin does not support conditional execution beyond digital signatures and time locks, requiring Lizz to enforce multi-party payment conditions without the use of smart contracts. To address these limitations, Lizz introduces a committee-based verification mechanism. This mechanism allows the committee to collectively approve state updates only when predefined conditions are met. Additionally, it enables honest members to generate penalty transactions against dishonest actors, deterring fraud and upholding the integrity of the protocol.

Our contributions are as follows:

1) We propose Lizz, the first protocol for constructing multi-party conditional payment channels on blockchain systems with limited scripting capabilities. Lizz introduces a user-participated honest-majority committee to handle adjudication and verification functions and employs a three-phase update process—state generation, condition fulfillment, and fulfillment verification—enabling secure and scalable multi-party payments with programmable conditions.

2) We introduced a proxy to support users joining and exiting the channel without rebuilding it. To join, a user establishes a TPC with the proxy outside the multi-party channel and synchronizes updates through conditional payments within the multi-party channel. To exit, a user makes a payment to the proxy within the multi-party channel, with the proxy's on-chain payment to the exiting user as a condition.

3) We provide a formal security analysis of Lizz within the UC framework, proving its security, robustness, and fairness under realistic adversarial settings.

4) We evaluate Lizz's cost by comparing it to SOTA non-Bitcoin-compatible multi-party schemes achieving the same payment goals, and to Bitcoin-compatible TPC schemes for efficiency (See Table 1). The results show that in a complex supply chain finance scenario with 96 users, the on-chain transaction size for opening and closing the channel is reduced by 77.14%, leading to a corresponding 77.14% cost saving. The average on-chain transaction size for a user joining and exiting is reduced by 98.1%. When consolidated and converted to USD, this results in a 77.56% cost saving. Additionally, Lizz demonstrates low latency and high scalability, with system throughput increasing by 218%, making it well-suited for multi-party conditional payments.

Table 1: Comparison of Different Multi-Party Payment Channel Protocols

Protocols	Flexibility[1]	Compatibility[2]	Conditionality[3]
MPC [14]	✓	×	×
MPC+ [12]	✓	×	×
MPCN [5]	✓	×	×
Garou [28]	✓	×	×
Magma [7]	✓	×	×
Lizz(ours)	✓	✓	✓

[1] Users join and exit the channel without rebuilding it.
[2] The channel can be deployed on non-Turing complete blockchain systems like Bitcoin.
[3] Payments within the channel can be conditioned on one or more real-world outcomes.

2 Relatex Work

2.1 Two-Party Payment Channel

A Two-party Payment Channel (TPC) is a mechanism that allows two participants to conduct multiple transactions off-chain, with only the initial and final states being recorded on the blockchain. This reduces the load on the blockchain, minimizes transaction fees, and enables near-instant transactions, enhancing efficiency and scalability.

Payment channels: Research on payment channels initially emerged on Bitcoin-compatible blockchain platforms [23], leading to various innovative protocols. For example, the BOLT [11] protocol introduced an anonymous TPC that aims to provide privacy protection while optimizing performance. Teechan [15] leveraged Trusted Execution Environments (TEEs) to enhance TPC performance, significantly reducing on-chain transactions and latency. Sprites [20] extended channel operations beyond payments to support arbitrary state transitions on the underlying blockchain, reducing payment confirmation time and increasing efficiency. Aumayr [1] proposed and formalized a new general channel concept that significantly expands the functionality of TPC, enabling more efficient and complex off-chain operations on cryptocurrencies like Bitcoin, and reducing communication complexity and on-chain overhead. However, using payment channels requires establishing channels between all parties involved in the transactions, resulting in high costs and limited scalability.

Payment channel networks: To enhance the scalabil-

ity of payment channels, researchers have proposed Payment Channel Networks (PCNs) [23]. PCNs connect multiple TPCs and enable indirect payments between parties through multiple intermediaries along the path, thus improving scalability. For example, Ersoy [6] implemented synchronous and asynchronous multi-hop payments for PCNs. Blitz [3] proposed a new multi-hop payment protocol that prevents malicious intermediaries from stealing funds using single-round transactions. SPRITE [21] enabled concurrent transactions in PCNs through offline path finding and online transaction processing, reducing message complexity. Podiatchev [22] analyzed the average downtime of channels in PCNs and proposed an optimized channel capacity distribution scheme to increase the minimum downtime and improve the payment success rate.

However, using Payment Channel Networks requires the active involvement of an intermediary for each payment between two users. This introduces additional costs, such as transaction fees to incentivize the intermediary. Moreover, unreliable intermediaries may compromise transaction security, potentially leading to privacy breaches or even loss of funds.

In contrast, our proposed payment channel solution, Lizz, eliminates the need for unreliable intermediaries to link multiple payment channels. Instead, all participants share a single payment channel, managed securely by a distributed committee that oversees the use of channel funds.

2.2 Multi-Party Payment Channel

Multi-party payment channels allow multiple participants to conduct many-to-one, one-to-many, or many-to-many payments off-chain, with only the initial and final states recorded on the blockchain. These channels are typically managed by smart contracts or administrators who coordinate participants and adjudicate disputes. For example, Studies [14] [12] [5] extended PCNs to support multi-party transactions within a single channel. MPC [14] is designed for scenarios where a central user receives payments from other nodes, handling multiple many-to-one payments simultaneously. MPC+ [12] optimizes MPC by allowing multiple users to exit the channel simultaneously, improving efficiency through binding strategies. MPCN [5] enhances efficiency by using a shared channel for proxy payments with an improved transaction optimization strategy. Another study [24] employs state channels and virtual channels for off-chain payments, introducing a global state channel contract that resolves payment disputes in constant time without blockchain involvement, ensuring intermediary nodes do not lose funds. However, these solutions fundamentally rely on the traditional TPC model for indirect payments, which is limited to point-to-point transactions. This approach may still incur significant costs in multi-party payment scenarios involving many-to-many transactions.

The works most closely related to ours are Garou [28] and Magma [7], which introduce a special role (leader or operator) to centralize the channel update process. In Magma, the operator is fixed upon creation, while in Garou, the leader is periodically elected. In both systems, the channel update process is executed in epochs. During each epoch, the payer sends transaction messages and corresponding signatures to the central entity, which coordinates the payments and broadcasts the new state. Because these systems require smart contracts to perform functions beyond payments, such as adjudicating disputes and managing users to join and exit the channel, they lack compatibility with non-Turing complete blockchain systems. Additionally, current multi-party solutions lack conditionality, as they do not support conditional payments.

3 Problem Statement

3.1 Preliminaries

3.1.1 Transactions in Supply Chain Scenarios

In supply chain finance, payments typically follow a one-to-many model, where a manufacturer or retailer distributes funds to multiple suppliers based on conditions such as delivery confirmation or contract fulfillment [16]. Unlike peer-to-peer payments, these transactions require structured mechanisms to ensure secure and efficient fund distribution. Many-to-many payments are rarely needed, as manufacturers typically procure materials from a single supplier or suppliers independently. Therefore, this paper focuses solely on the one-to-many model, which aligns with the prevalent payment structure in supply chain scenarios.

3.1.2 Pay-to-Script-Hash (P2SH) and Script_sig

In Bitcoin's UTXO model, *Pay-to-Script-Hash* (P2SH) and *Script_sig* are closely interconnected components that facilitate secure and flexible transaction validation. P2SH allows funds to be sent to the hash of a script, rather than directly to a public key hash. This script, referred to as the *redeem script*, specifies arbitrary spending conditions such as multi-signature requirements or time locks. The redeem script itself is not included in the transaction output but is instead provided by the spender when the funds are later spent. This design shifts the responsibility of supplying the spending conditions from the sender to the redeemer, reducing transaction size and complexity for the sender. When redeeming the funds, the transaction input includes a *Script_sig*, which acts as the unlocking script. In the context of P2SH, Script_sig typically contains the necessary data to satisfy the conditions defined in the redeem script—such as digital signatures—along with the redeem script itself. During trans-

action verification, Script_sig is combined with the locking script of the referenced UTXO, and the complete script is executed to ensure all spending conditions are met. By deferring script complexity to the redemption phase, P2SH and Script_sig together enable expressive, condition-based spending mechanisms without sacrificing network efficiency or scalability.

3.2 Threat Model

In the Lizz architecture, as shown in Figure 1, users initiate off-chain payment requests with proof that conditions are met, while a committee—a selected subset of users—verifies these proofs and updates the channel state to ensure correct and efficient transaction execution. Users can join or exit the channel through a proxy within the channel.

Committee. The committee operates under an honest-majority assumption to ensure reliable coordination of payment channels. Although advanced cryptographic techniques such as homomorphic encryption and zero-knowledge proofs can reduce trust assumptions [25], they typically incur high computational overhead, making them impractical for Lizz's real-time coordination needs. In contrast, the honest-majority assumption is more feasible in environments with reputable industry entities(see Sec.7), such as those in supply chain finance. Potential threats include collusion among committee members to forge transactions and denial-of-service attacks.

Users. Users within the channel interact with the committee to execute off-chain transactions by submitting payment requests along with required proofs. While generally honest, some users may engage in malicious behaviors such as double-spending attacks, uploading outdated transactions, non-cooperation, or attempting to defraud other participants by submitting invalid proofs.

Proxy. A proxy is an untrusted role created by users within the channel, allowing users to join or exit the channel with proxy assistance in updates. Off-channel transactions are verified by the committee to ensure payment conditions are met. Potential threats include non-cooperation and proof forgery.

3.3 Security and Performance Goals

As previously mentioned, Lizz's goals aim to achieve *security, efficiency, compatibility, conditionality, and flexibility.*

Security: Honest users are protected from any loss of coins, and the protocol ensures that malicious users cannot disrupt the channel's operation.

Efficiency: Users can make payments to multiple users within the channel simultaneously without intermediaries.

Compatibility: The channel can be deployed on non-Turing complete blockchain systems, such as Bitcoin.

Conditionality: Payments within the channel can be conditioned on one or more real-world outcomes.

Flexibility: Users can join the channel without interrupting its operations and exit the channel without affecting the interests of other users.

4 System Design

We propose Lizz, a Bitcoin-compatible multi-party payment channel protocol that supports conditional payments. This section provides a detailed description of the system workflow and the implementation of each procedure. For the sake of clarity, we omit the details of the committee's threshold signature process for each round of communication, as these are thoroughly detailed in [25]. The signature technology utilized is Bitcoin's official threshold signature scheme [9].

4.1 Overview

As shown in Figure 1, there are five procedures during the lifetime of Lizz, *opening*, *update*, *closing*, *joining*, and *exiting*.

Channel Opening. After the committee is selected, it defines the corresponding redeem script and generates a P2SH address. Users then create funding transactions to this P2SH address and submit them to the blockchain.

Channel Update. The channel state, reflecting the users' balances, is updated based on transactions made by the channel users. In this protocol, the update is initiated by a user creating a transaction. The receiver signs the transaction and provides proof of condition fulfillment to the committee. After the committee verifies the conditions and performs a threshold signature, the complete transaction is sent back to the receiver.

Channel Closing. The receiver who provided the signature in the latest update submits the latest channel state to close the channel, ensuring that balances are refunded to users based on this final state. Once the closure is complete, all parties are notified, and the channel is terminated.

Channel Joining. A user can request an existing channel user to act as a proxy for joining the channel. Together, the user and the proxy establish a TPC outside the channel.

Channel Exiting. A user can request an existing channel user to act as a proxy. The user transfers his channel balance to the proxy, which then generates an equivalent transaction on the blockchain to facilitate the user's exit from the channel.

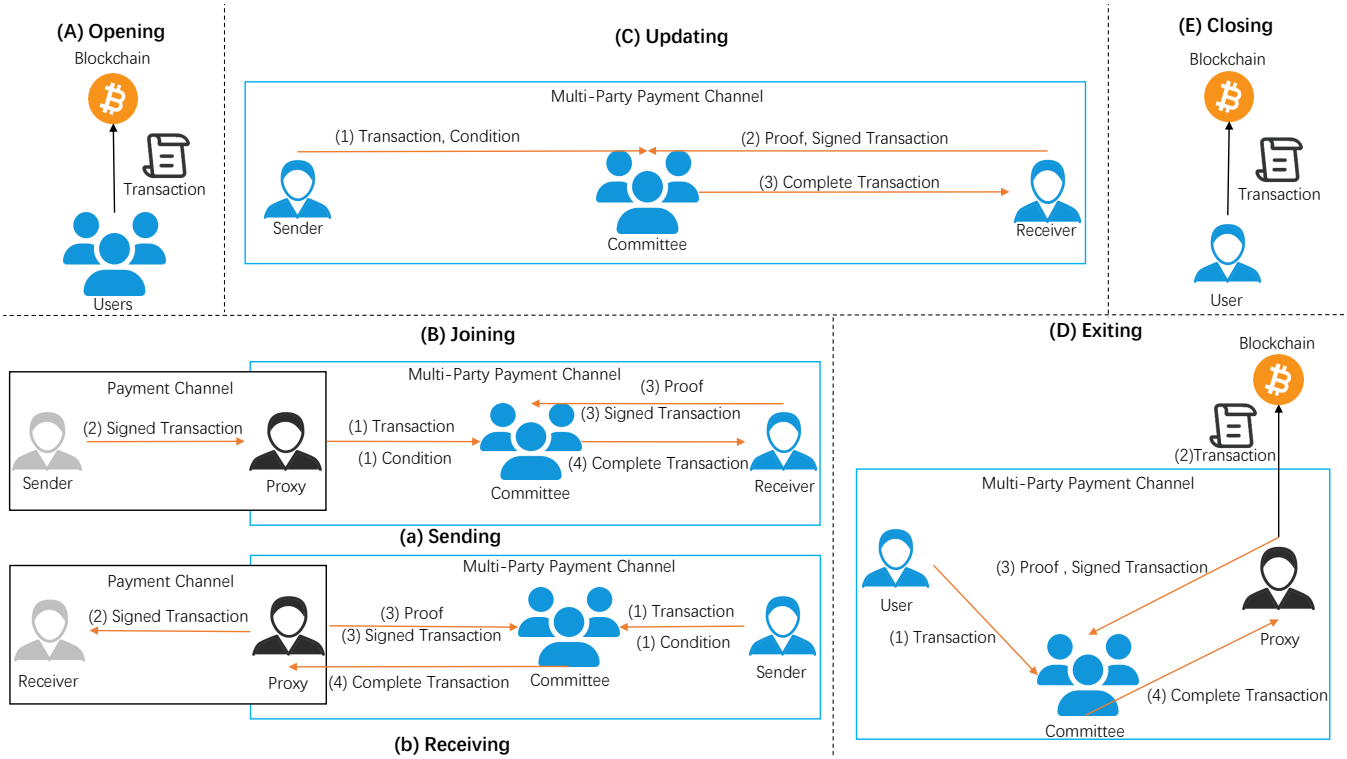

Figure 1: The Lizz architecture. Orange lines indicate off-chain in-channel messages. Black lines indicate on-chain messages. Blue boxes represent Lizz, and black boxes represent two-party payment channels. (A) Users submit transactions on-chain to fund the channel. (B.a) When a user joins as a sender, the proxy makes payments in the multi-party channel based on the user's payment in the two-party channel. (B.b) When a user joins as a receiver, the sender pays the proxy in the multi-party channel based on the proxy's payment to the user in the two-party channel. (C) During a channel update, the sender creates a transaction and defines the payment conditions. Any receiver can sign the transaction, and all receivers provide proof of condition fulfillment. The committee then confirms the payment conditions and performs a threshold signature, sending the complete transaction to the signing receiver. (D) When a user exits, the user pays the proxy in the channel, conditional on the proxy making an on-chain payment to the exiting user. (E) The channel is closed when the most recent receiver submits the complete transaction on-chain.

4.2 Channel Opening

Opening a channel requires users to fund the channel and define unlocking conditions. A subset of users forms an honest-majority committee with a threshold t, preventing single points of failure or corruption. Typically, the threshold is set to more than half of the committee members to prevent 51% attacks, assuming an honest majority. The committee collects all user public keys to construct the redeem script S_{redeem}, which defines the unlocking conditions as the signatures of any two users and the threshold signatures of the committee or after a specified time T. A P2SH (Pay-to-Script-Hash) address is generated from S_{redeem}. This setup ensures that the channel operates with minimal modules, preventing failure due to user inactivity. If all users become uncooperative, the committee can generate a transaction and apply the threshold signature after time T to return users' funds and close the channel.

Users then send funding transactions to this P2SH address, locking their maximum usable amount α_i in the channel. The committee creates a state table S, recording each user's balance α_i based on each user's funding transaction Tx_{f_i}, and broadcasts it to all users. This setup helps prevent double-spending attacks and state tampering.

4.3 Channel Updating

The channel update process is explained in three phases: state generation, condition fulfillment, and fulfillment verification.

4.3.1 State Generation

In the state generation phase, the sender creates a transaction, sets the corresponding payment conditions, and signs the transaction. The sender constructs a MIMO (Multiple-Input Multiple-Output) transaction Tx_s, with inputs from Tx_{f_i}, and outputs representing the new states of all users, including a time lock and the signature Sig_s. For the first update, a time lock of duration T is set, with subsequent updates having a decreased time lock to prevent uploading outdated states. If a user attempts to perform a replay attack by uploading an outdated state, the committee can generate a transaction with a shorter

Algorithm 1: Condition Fulfillment Phase

Input: Broadcast message from P_s
Output: Signed transaction Tx_s, proof π_i, signature list Sig_r

P_c verifies P_s's payment capability using S;
Broadcast verification result to P_r;
if *P_s's capability confirmed* **then**
  P_r sends willingness W_i to P_c;
  P_c confirms willingness and broadcasts Sig_s, Tx_s, M_{con} to P_r;
One P_{r_i} generates proof π_i and signs Tx_s to get Sig_r;
Broadcast Sig_r and π_i to P_c;
for *each other P_{r_i}* **do**
  Broadcast proof π_i to P_c;
if *any P_{r_i} fails to meet conditions* **then**
  P_c generates a new transaction to penalize P_{r_i};
  Redistribute balance to other honest participants;

time lock to prevent malicious behavior. After constructing and signing Tx_s, the signed transaction, along with the payment conditions M_{con}, is broadcast to the committee.

4.3.2 Condition Fulfillment

In the condition fulfillment phase, the committee first verifies the sender's ability to pay using the state table S. Upon confirmation, the receivers send their willingness to fulfill the payment conditions. The committee confirms their willingness and broadcasts the transaction and conditions to all receivers. Each receiver generates proof π_i and signs the transaction Tx_s to get Sig_r, then broadcasts this to the committee, as shown in Algorithm 1. If any receiver fails to meet the conditions, the committee generates a new transaction to penalize the receiver and redistribute their balance to other honest participants.

4.3.3 Fulfillment Verification

In the fulfillment verification phase, the committee verifies the payment conditions using the proof π_i provided by the receivers. If the conditions are met, the committee applies a threshold signature Sig_{c_i} to the transaction Tx_r and broadcasts it to the remaining committee members. The last committee member constructs the scriptSig and includes it in Tx_s. The updated transaction Tx_c is then provided to the receiver who provided the signature.

4.4 Channel Closing

Closing a channel requires the most recently updated receiver to submit the complete transaction Tx_c to the blockchain within time T. If this is not done within the specified time, the committee generates a transaction

Algorithm 2: Joining the Channel

Input: External user P_j, Proxy P_p, Receiver P_r, Committee P_c
Output: Updated channel with P_j joined

P_j establishes TPC with P_p
TPC Payments as Condition $\mathbf{M}_{con}$
if *P_j is sender* **then**
  P_p generates and signs transaction Tx_s to pay P_r, obtaining Sig_s
  P_p broadcasts Tx_s, Sig_s, and M_{con} to P_c
  P_j generates equivalent payment transaction Tx_t in TPC, signs to obtain Sig_t, and sends to P_p
  P_r expresses willingness to fulfill conditions after receiving payment proof from P_j
  if *Verification successful by P_c* **then**
    P_c broadcasts Tx_s, Sig_s, and M_{con} to P_r
    P_j provides proof π_{pay} generated from Tx_t to P_r
    P_r signs Tx_s to obtain Tx_r, fulfills conditions, and generates proof π_{con}
    P_r broadcasts Sig_r and proof π containing π_{con} and π_{pay} to P_c
    if *Verification successful by P_c* **then**
      P_c applies threshold signature to generate complete transaction Tx_c
      P_c sends Tx_c to P_r
if *P_j is receiver* **then**
  P_s generates and signs transaction Tx_s to pay P_p, obtaining Sig_s
  P_s broadcasts Tx_s, Sig_s, and M_{con} to P_c
  P_p generates equivalent payment transaction Tx_t in TPC, signs to obtain Tx_t, and sends to P_j
  if *Verification successful by P_c* **then**
    P_c broadcasts Tx_s, Sig_s, and M_{con} to P_p
    P_p signs Tx_s to obtain Sig_r and provides proof π containing π_{con} and π_{pay} to P_c
    if *Verification successful by P_c* **then**
      P_c applies threshold signature to generate complete transaction Tx_c
      P_c sends Tx_c to P_p

Tx_{close} based on the latest state table S, signs it, and submits it to the blockchain. If a user attempts to submit an outdated state, the transaction information can identify the malicious user. The time lock for each update is decremented, allowing the committee to submit a punishment transaction with a shorter time lock, redistributing the malicious user's balance to other honest participants.

4.5 User Joining

To join the channel, an external user establishes a two-party channel (TPC) with a proxy already in the channel. The TPC payment acts as the conditional value M_{con} for channel updates, ensuring synchronization and preventing unilateral fund theft, as shown in Algorithm 2.

As a sender, the proxy signs and broadcasts a transaction Tx_s, along with Sig_s and M_{con}, to the committee. The user then issues a corresponding TPC transaction Tx_t to the proxy. After verifying the user's capability and intent, the committee forwards the transaction to the receiver. Upon receiving the user's proof π_{pay}, the receiver signs, and the committee assembles the final transaction Tx_c.

As a receiver, a sender in the channel signs Tx_s to the proxy, who appends its signature and sends a proof to the committee. The user fulfills the payment condition and provides proof π_{con}. After verification, the proxy signs to produce Sig_r, and the committee generates the complete transaction Tx_c.

This structured process enables secure and verifiable channel entry while preserving consistency through conditional payments and committee oversight.

4.6 User Exiting

Exiting the channel requires the exiting user to transfer their balance to the proxy, on the condition that the proxy sends an equivalent transaction to the exiting user outside the channel. The exiting user creates and signs the transaction Tx_s to pay the proxy, and broadcasts the transaction along with the conditions to the committee. The committee verifies the payment capability and fulfillment of the conditions, then sends the signed transaction to the proxy. The proxy submits the equivalent exit transaction $\mathrm{Tx}_{\mathrm{exit}}$ on-chain for the exiting user. The proxy provides proof π to the committee, which, upon verification, generates the complete transaction Tx_c and sends it to the proxy.

5 Security Analysis

5.1 Ideal Functionality $F_{\mathbf{Lizz}}$

The ideal functionality F_{Lizz} acts as a trusted entity that correctly handles conditional payments between users without requiring explicit blockchain operations. It is defined as follows:

Transaction Initialization: Upon receiving a request from a sender S to send amount to receiver R under condition C, the functionality stores the tuple (S, R, amount, C) without executing the payment immediately.

Condition Verification: If R provides proof P that condition C is satisfied, F_{Lizz} verifies P. If valid, it updates the state to mark the transaction as executable.

Payment Execution: Once P is verified, F_{Lizz} transfers amount from S to R, ensuring correct state tracking.

Fairness Guarantee: If R refuses to submit proof, the transaction is automatically canceled after a timeout T, preventing the sender's funds from being locked.

Double-Spending Prevention: F_{Lizz} maintains a state table ensuring that a sender cannot initiate two conflicting transactions using the same funds.

This ideal functionality ensures fairness, consistency, and security without allowing attacks such as double spending or non-cooperation.

5.2 Real-World Protocol $\pi_{\mathbf{Lizz}}$

The real-world Lizz protocol utilizes a committee of validators and threshold signatures to ensure that payments are executed correctly. The protocol proceeds as follows:

Transaction Initialization: S submits a payment request $T = (S, R, \text{amount}, C)$ to the committee CMT, which maintains a global state table $StateTable$.

Condition Verification: If R submits proof P, the committee verifies it: If valid, the committee signs the transaction using threshold signatures and returns the signed transaction T' to R. If invalid, the transaction is rejected.

Payment Execution: R submits T' to the blockchain, finalizing the transfer.

Fairness Guarantee: If R does not submit T', the committee can force execution after timeout T.

Double-Spending Prevention: The committee ensures that no two conflicting transactions from S are signed.

The key challenge is proving that π_{Lizz} correctly simulates F_{Lizz} under adversarial conditions.

5.3 Simulator Construction $S_{\mathbf{Lizz}}$

To prove UC security, we construct a simulator S_{Lizz} that mimics the real-world execution while interacting with F_{Lizz}. The simulator performs the following steps:

Transaction Handling: When S submits a transaction, S_{Lizz} forwards it to F_{Lizz}, ensuring identical behavior.

Condition Verification: When R submits proof P, S_{Lizz} checks with F_{Lizz}. If valid, S_{Lizz} generates a simulated threshold signature, indistinguishable from a real signature.

Fairness and Execution: If R fails to submit T', S_{Lizz} enforces timeout execution, consistent with F_{Lizz}.

Anti-Double-Spending Measures: If S attempts to submit two conflicting transactions, S_{Lizz} rejects them, replicating F_{Lizz}'s response.

The key goal is to show that an adversary interacting with S_{Lizz} cannot distinguish it from the real-world protocol π_{Lizz}.

5.4 UC Security Proof

To complete the proof, we must demonstrate that π_{Lizz} UC-emulates F_{Lizz}:

Theorem 1 (UC-Security of Lizz Protocol)**.** *For any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a simulator S_{Lizz} such that no environment $\mathcal{Z}$ can distinguish between interactions with π_{Lizz} and F_{Lizz}, except with negligible probability.*

Proof Sketch. We argue that for all possible attacks $\mathcal{A}$ could launch in π_{Lizz}, there exists a corresponding response in F_{Lizz}:

Correctness: S_{Lizz} correctly forwards messages between parties, ensuring transactions occur exactly as they would in F_{Lizz}.

Indistinguishability: Threshold signatures and committee responses in S_{Lizz} are indistinguishable from real committee interactions in π_{Lizz}.

Adversarial Attacks: If $\mathcal{A}$ attempts a double-spending attack, S_{Lizz} prevents it using the same consistency checks as F_{Lizz}. If $\mathcal{A}$ tries to manipulate committee behavior, S_{Lizz} enforces the same honest-majority rule as π_{Lizz}.

Since $\mathcal{Z}$ cannot distinguish between π_{Lizz} and S_{Lizz}, the UC security definition is satisfied, proving that the Lizz protocol is UC-secure. □

We have demonstrated that the Lizz protocol achieves UC security by constructing an ideal functionality, defining a real-world protocol, and proving its indistinguishability through a well-structured simulator. This guarantees protection against double-spending, non-cooperation, and committee collusion attacks. The protocol is also efficient, compatible with Bitcoin, conditional, and flexible, ensuring its real-world applicability and robustness.

6 Implementation and Evaluation

6.1 Baselines

In this section, we introduce the baseline schemes used for comparison in our experiments. These baselines were selected to represent common approaches in multi-party payment channels, each highlighting a different aspect of payment channel performance. (1) Since PCNs [23] rely on intermediaries, they do not meet the efficiency criteria defined in this paper. We use Bitcoin-compatible TPC [1] as a baseline, a widely adopted two-party solution that benchmarks Bitcoin compatibility and performance. Since TPC does not support multi-party payments, it highlights Lizz's efficiency in enabling direct payments to multiple participants without intermediaries. (2) For smart contract-driven solutions, we compare MPC [14] and MPC+ [12], which extend PCNs to support many-to-one payments. These solutions highlight Lizz's superior efficiency in handling multi-party transactions without intermediaries. (3) Lastly, we compare Magma [7], which extends two-party channels into robust multi-party channels for many-to-many payments. Magma helps emphasize Lizz's scalability, lower deployment costs, and better Bitcoin compatibility.

6.2 Implementation

We used a Python script to call the RPC interface on Bitcoin regtest to create multiple address accounts as users and obtain UTXOs. We then used the python-bitcoin-utils library and the Bitcoin Script language to create and sign transactions. Since committees perform the same functions as users, they are represented by the user accounts. The first few users acted as committees, while the first user in each layer served as a proxy. We have open-sourced Lizz at [26].

6.3 Setup

In order to closely mimic real-world conditions, we implemented the prototype with supply chain scenarios on an AliCloud server equipped with an Intel(R) Xeon(R) Platinum 8-core processor and 16GB of RAM, and conducted experiments with varying numbers of users. The entire scenario was divided into three layers: the first layer consisted of raw material suppliers and parts suppliers, the second layer included manufacturers, and the third layer comprised sellers. Each layer had distinct roles, with all nodes in the upper layers needing to pay for purchases based on the quantity of goods.

The complete payment process was divided into two rounds. In the first round, each user in the second layer paid the respective users in the first layer. The second round was similar but involved the third layer paying the second layer. Each payment round included multiple one-to-many payments, where the initiator created the transaction, signed it with a payment condition, and broadcasted it. The receiver, upon fulfilling the condition, broadcasted the signed transaction and proof of condition fulfillment to a committee of 3 users, with a threshold of 2. The committee members verified the condition fulfillment and performed a threshold signature to send the transaction to the receiver who provided the signature. We omitted the operations of the committee verifying the payment ability and condition fulfillment, as these were

Table 2: Comparison of Deployment Costs Across SOTA Multi-Party Payment Channel Schemes. n denotes the number of users. Total Cost represents the overall monetary cost (in USD) required to establish channels to achieve the same payment objectives.

Behavior	On-chain Tx	Total Cost/USD	Complexity
create TPCs	$\frac{2n^2}{9}$	$0.84n^2$	$O(n^2)$
create MPCs	$\frac{2n}{3} + \frac{2n^2}{9}$	$64.6n + 0.569n^2$	$O(n^2)$
n nodes join MPCs	n	2.34n	$O(n)$
create MPC+s	$\frac{2n}{3} + \frac{2n^2}{9}$	$66.3n + 0.569n^2$	$O(n^2)$
n nodes join MPC+s	n	2.34n	$O(n)$
create Magma	$n + 1$	$1.36n + 99.10$	$O(n)$
n nodes join Magma	n	1.36n	$O(n)$
create Lizz	1	$1.78n + 1.27$	$O(1)$
n nodes join Lizz	n	3.77n	$O(n)$

simple conditional judgments. Also, we omitted the consensus process, as consensus was not the main purpose of this paper.

6.4 Evaluation

We analyzed the complexity of different operations for on-chain transactions in Table 2. The costs for OPEN and CLOSE operations, as well as UPDATE, JOIN, and EXIT operations under simple realistic conditions are detailed in Figure 2, 3 and Figure 4, respectively. Finally, Figure 5 presented the total cost of Lizz compared to TPC for achieving the same payment goal with varying numbers of users. These costs were compared with the payment channel costs on both sides required by the TPC scheme for the same payment purpose. The cost in USD was calculated using the following parameters: The price of Bitcoin was \$61,731 per bitcoin. The communication cost was based on the AliCloud server rate of \$0.11 per GB. The computation cost was based on the AliCloud server rate of \$0.019 per hour. At the time of writing, the transaction cost per byte for a Bitcoin transaction to be accepted by the next block was 15.922 Satoshis.

6.4.1 Comparison with Other Multi-party Payment Channel Solutions

In Table 2, we analyze the number of on-chain transactions, costs, and complexities involved in opening payment channels for Lizz and other schemes, specifically considering the number of channels needed to achieve the payment goals in the three-tier supply chain scenario outlined in SETUP. Since the costs of off-chain updates are negligible compared to on-chain costs, and the closing costs are similar to opening costs, we focus only on the costs associated with opening the channels.

We observe that Bitcoin-compatible TPC schemes, along with MPC and MPC+, require quadratic numbers of channels and associated costs due to their lack of support for many-to-one and many-to-many payments. While Magma also requires users to join after the channel is opened, its support for many-to-many payments results in on-chain transaction complexity proportional to the number of users (n). Lizz, though not supporting many-to-many payments, converts these into multiple one-to-many payments, requiring only a single channel.

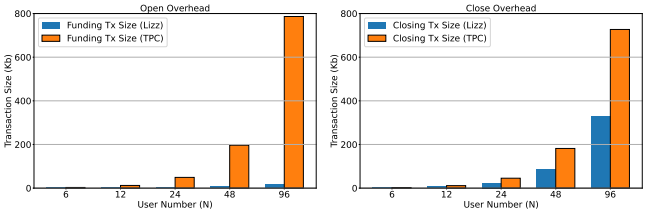

Figure 2: The byte size of on-chain transactions for opening and closing the channel for Lizz and TPC were compared for different numbers of users

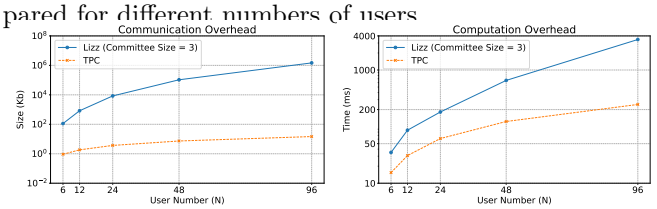

Figure 3: Communication overhead for Lizz and TPC when updating the complete payment process with different numbers of users, and the computation overhead required for creating and signing transactions.

The main cost difference between Lizz and Magma lies in the operator's expense for opening the channel. Magma incurs higher initial costs that decrease as the user base grows, whereas Lizz's costs increase due to Bitcoin's limitations, specifically needing to include all users' public keys in the script. Magma's costs match Lizz's when the user base reaches 223, but supply chain scenarios typically do not require this many participants, making Lizz more suitable for such environments.

Regarding joining the channel, the costs for MPC+ and MPC are identical, while Magma's cost is the lowest. Since these schemes create channels first and users join gradually, Lizz incurs slightly higher costs when adding users because it includes all participants when the channel is opened.

6.4.2 Comparison with Bitcoin-Compatible Two-Party Payment Channels

Open and close channel: We tested the size of the funding transactions for opening a channel and the size of the closing channel transactions, comparing them to the TPC scheme, as shown in Figure 2. We observed that the cost of opening and closing a channel increased as the number of users grew. Notably, closing transactions grew faster than funding transactions because the redeem script for the channel appeared only once in the output of a funding transaction. However, when closing the channel, the output for each user contained a complete redeem script. In contrast, the TPC showed a small difference in the size of the funding and closing channel transactions because there were only two users. As the number of participating users increased, the number of TPC required grew quadratically. For example, with 96 users, the total transaction sizes for the Lizz and TPC schemes to open and close the channel were 346.09 KB and 1513.47 KB, respectively, saving 77.13% of the transaction size.

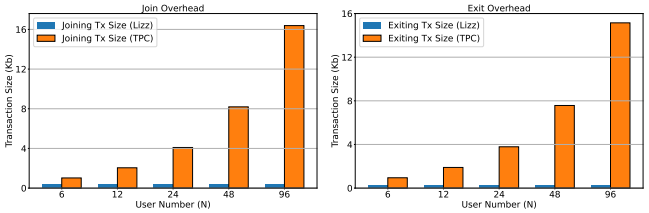

Figure 4: Average on-chain transaction byte sizes for Lizz and TPC for a user joining and exiting with different numbers of users.

Update channel: We tested the overhead (communication and computation) associated with a channel update for a complete payment process using a simple reality condition as the payment condition and compared it with the TPC scheme. The results, shown in Figure 3, indicate that both communication and computation overhead increase with the number of users. For example, with 96 users, the communication overhead for Lizz was 1.47GB and 14.72KB for TPC, while the computation overhead was 3455.67ms for Lizz and 246.28ms for TPC during the creation and signing of transactions. The costs in USD were of the order of 10^{-1} and 10^{-5}, which were not displayed in the figure due to their insignificance. Although the channel update cost was higher than the TPC scheme, the significant savings in on-chain costs made the off-chain overhead negligible, as demonstrated in Figure 5.

Join and exit channel: We tested the size of the on-chain transactions generated by a user joining and exiting to compare with the TPC scheme, and the results were shown in Figure 4. Since the TPC scheme was inconsistent in terms of the number of TPCs that needed to be activated or deactivated to handle the joining and exiting scenarios of each layer of users, we averaged the number by probability to account for the cost. The results showed that the number of Lizz users was independent of the on-chain cost of joining and exiting because joining required activating a TPC with the proxy outside the channel while exiting required sending an on-chain transaction from the proxy within the channel. In contrast, for TPC, the cost of joining and exiting a user increased with the overall number of users, since the number of TPCs that needed to be opened and closed was of order O(n). For example, with 96 users, the Lizz and TPC schemes had a user join and exit channel transaction size sum of 0.6KB and 31.53KB, respectively, saving 98.1% of the transaction size.

We summarized the above on-chain and off-chain costs and compared Lizz with TPC, as shown in Figure 5. As mentioned earlier, the savings in on-chain costs of Lizz for the TPC scheme were much larger than the increase in off-chain costs. Taking 96 users as an example, the total cost of Lizz and TPC was $3408 and $15188 respectively, which saved 77.56%.

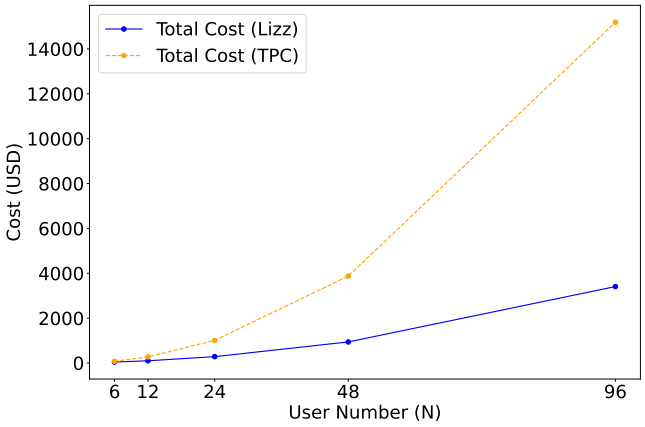

Figure 5: Total cost of on-chain and off-chain for Lizz and TPC with different numbers of users.

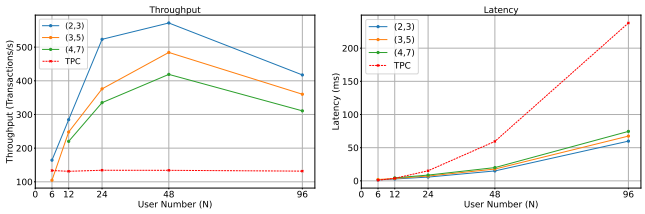

Figure 6: Throughput and communication latency of Lizz and TPC under different numbers of users and varying committee sizes.

6.5 System Throughput and Communication Latency

We tested the throughput and communication latency under different numbers of users, varying committee sizes, and a 10ms communication delay, comparing these results with the TPC scheme to achieve the same payment objectives, as shown in Figure 6. The more channel users there were, the longer the communication delay. This was because the number of channel updates increased with the number of users. Larger committee sizes also led to longer communication delays and lower system throughput, as more communication rounds and longer computation times were required. However, it was noteworthy that with up to 48 channel users, the system throughput increased with the number of users. This was because the in-channel payments were one-to-many, and more users meant more transactions per update. When the number of users reached 96, the system throughput decreased due to the larger transaction size, which included 96 inputs and outputs, leading to longer computation times. With 96 users and a committee size of 3, Lizz achieved a throughput of 417 transactions per second and a latency of 60ms, compared to the TPC scheme's 131 transactions per second and 238ms. This represented a 218% increase in throughput and a 74.79% reduction in latency.

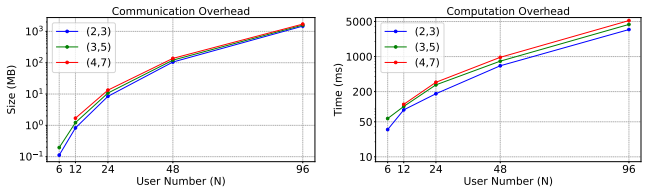

Figure 7: Impact of committee size on the overhead of channel updates for different numbers of users.

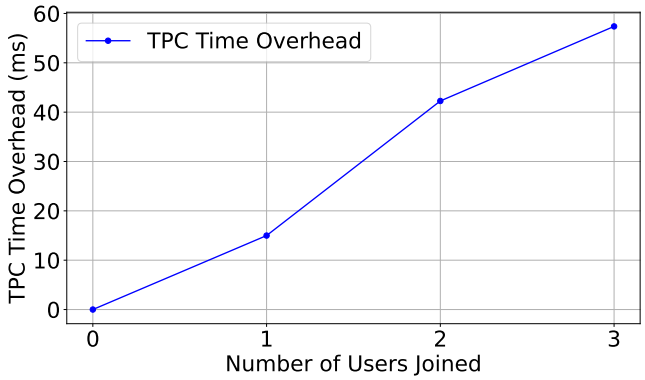

Figure 8: The impact of different numbers of joining users on time overhead.

6.6 Impact of System Parameters

6.6.1 Impact of Increasing Committee

We tested the impact of different committee sizes on the overhead of channel updates, as shown in Figure 7. The overall increase was not significant. For example, with 96 users, increasing the committee size from 3 to 7 resulted in a 16.02% increase in communication overhead and a 51.11% increase in computation overhead. This modest increase occurred because the committee added signatures to the signature list and broadcasted only the DER-encoded signatures.

6.6.2 Impact of Increasing User Join

We had three users join the first, second, and third layers, respectively, and complete a round of payments. Since theoretically, user joining is independent of channel user size, we tested the time overhead required in the TPC for creating and updating transactions after different numbers of users had joined when there were 6 channel users, including 3 committee members. The results are shown in Figure 8. The figure showed that the overhead increase for the first and third users was almost the same, while the increase for the second user was more significant. This was because users in the first and third layers only needed to pay or receive payments, whereas the second-layer user needed to receive payments from the first layer and make payments to the third layer. Ultimately, the additional time overhead in the TPC for a user joining the first or third layer averaged 15.05ms, while for a user joining the second layer, it averaged 27.28ms.

7 Discussion

In the future, we believe there are two directions:

Honest majority assumption in the committee. Since Lizz relies on an honest majority within the committee to enhance security, reducing trust assumptions is essential. In real-world scenarios, such as supply chains, committees can be organized by industry unions or associations, whose authority helps ensure a baseline level of trustworthiness among committee members. Technologies like Trusted Execution Environments (TEEs) can enhance security by requiring committee members to run the Lizz protocol within a TEE, providing verifiable proof of execution integrity and making actions tamper-resistant. Cryptographic techniques [10], such as homomorphic encryption and zero-knowledge proofs, can further reduce trust requirements in threshold signatures. For committees with more than three members, PBFTs [17,18] can also lower trust assumptions for secure operation.

Privacy. Since Lizz uses a committee to verify payment capabilities and conditions, this inevitably causes privacy issues. In future work, Lizz will attempt to replace the verification process with non-interactive zero-knowledge proofs to protect the privacy of the parties involved in the transaction.

8 Conclusion

In this paper, we propose Lizz, a Bitcoin-compatible multi-party conditional payment channel that enhances efficiency and security in decentralized payments. Lizz employs a user-managed committee to track balances securely and resolve disputes. It enables conditional payments via threshold signatures and allows honest users to penalize dishonest actors, reinforcing fraud resistance.

A proxy role facilitates flexible channel entry and exit. Theoretical analysis under the UC framework confirms Lizz's security, efficiency, compatibility, conditionality, and flexibility. Experiments in a 96-user supply chain finance scenario show that Lizz reduces deployment costs compared to SOTA non-Bitcoin-compatible schemes and lowers overhead by 77.14% compared to Bitcoin-compatible TPCs. Lizz also achieves low latency and high scalability, increasing system throughput from 131 tx/s to 417 tx/s—a 218% improvement—making it well-suited for multi-party conditional payments.

Acknowledgments

This work was supported in part by the Key Program of Nature Science Foundation of Zhejiang Province under Grant LZ24F020007 , in part by the National Nature Science Foundation of China under Grant 62072407, in part by the "Leading Goose Project Plan" of Zhejiang

Province under Grant 2022C01086, Grant 2022C03139, and in part by the National Key R&D Program of China under Grant 2022YFB2701400.

References

[1] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. M. Sanchez, and S. Riahi, "Generalized bitcoin - compatible channels," in *IACR Cryptol. ePrint Arch.*, p. 476, 2020.

[2] L. Aumayr, M. Maffei, O. Ersoy, A. Erwig, S. Faust, S. Riahi, K. Hostáková, and P. Moreno Sanchez, "Bitcoin - compatible virtual channels," in *2021 IEEE Symposium on Security and Privacy (SP)*, p. 901–918. IEEE, Manhattan, New York, U.S., 2021.

[3] L. Aumayr, P. M. Sanchez, A. Kate, and M. Maffei, "Blitz: Secure $$multi - hop$$ payments without $$two - phase$$ commits," in *30th USENIX Security Symposium (USENIX Security 21)*, p. 4043–4060. USENIX, Berkeley, CA., 2021.

[4] Binance. "Binance - buy & sell bitcoin, ethereum, and more with trust,".

[5] Y. Chen, X. Li, J. Zhang, and H. Bi, "Multi-party payment channel network based on smart contract," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, p. 4847–4857, 2022.

[6] O. Ersoy, J. Decouchant, S. P. Kumble, and S. Roos, "Syncpcn/psyncpcn: Payment channel networks without blockchain synchrony," in *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, p. 16–29, Cambridge, MA, USA, 2022. Association for Computing Machinery, New York, NY, United States.

[7] Z. Ge, Y. Zhang, Y. Long, and D. Gu, "Magma: Robust and flexible multi-party payment channel," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, p. 5024–5042, 2023.

[8] L. M. Gelsomino, R. Mangiaracina, A. Perego, and A. Tumino, "Supply chain finance: a literature review," *International Journal of Physical Distribution & Logistics Management*, vol. 46, no. 4, 2016.

[9] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ecdsa with fast trustless setup," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, p. 1179–1194, 2018.

[10] R. Gennaro, S. Goldfeder, and A. Narayanan, "Threshold - optimal dsa/ecdsa signatures and an application to bitcoin wallet security," in *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19 - 22, 2016. Proceedings 14*, p. 156–174. Springer, 2016.

[11] 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*, p. 473–489, Dallas, Texas, USA, 2017. Association for Computing Machinery, New York, NY, United States.

[12] L. Huang, H. Lei, and L. Wang, "Mpc+: Secure, compatible and efficient off-blockchain multi-node payment channel," *IEEE Transactions on Network and Service Management*, 2024.

[13] X. Jia, Z. Yu, J. Shao, R. Lu, G. Wei, and Z. Liu, "Cross-chain virtual payment channels," *IEEE Transactions on Information Forensics and Security*, 2023.

[14] H. Lei, L. Huang, L. Wang, and J. Chen, "Mpc: Multi - node payment channel for off - chain transactions," in *ICC 2022 - IEEE International Conference on Communications*, p. 4733–4738, Seoul, Korea, 2022. IEEE, Manhattan, New York, U.S.

[15] J. Lind, I. Eyal, P. Pietzuch, and E. Sirer, "Teechan: Payment channels using trusted execution environments," *arXiv preprint arXiv:1612.07766*, 2016.

[16] F. Liu, M. Fang, K. Park, and X. Chen, "Supply chain finance, performance and risk: how do smes adjust their buyer - supplier relationship for competitiveness," *Journal of Competitiveness*, 2021.

[17] J. Liu, X. Deng, W. Li, and K. Li, "Cg - pbft: an efficient pbft algorithm based on credit grouping," *Journal of Cloud Computing*, vol. 13, no. 1, p. 74, 2024.

[18] H. Luo, G. Sun, H. Yu, B. Lei, and M. Guizani, "An energy - efficient wireless blockchain sharding scheme for pbft consensus," *IEEE Transactions on Network Science and Engineering*, 2024.

[19] V. Madathil, S. A. Thyagarajan, D. Vasilopoulos, L. Fournier, G. Malavolta, and P. Moreno-Sanchez, "Cryptographic oracle-based conditional payments," *Cryptology ePrint Archive*, 2022.

[20] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and state channels: Payment networks that go faster than lightning," in *International conference on financial cryptography and data security*, p. 508–526, St. Kitts, Saint Kitts and Nevis, 199. Springer - Verlag, Berlin.

[21] G. Panwar, R. Vishwanathan, G. Torres, and S. Misra, "Sprite: Secure and private routing in payment channel networks," *Cryptology ePrint Archive*, 2024.

[22] Y. Podiatchev, A. Orda, and O. Rottenstreich, "Survivable payment channel networks," in *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, p. 479–487, Bengaluru, India, 2024. IEEE, Manhattan, New York, U.S.

[23] J. Poon and T. Dryja. "The bitcoin lightning network: scalable off - chain instant payments (2016),", 2016.

[24] S. S. Sahoo, A. R. Menon, and V. K. Chaurasiya, "Blockchain based n - party virtual payment model with concurrent execution," *Arabian Journal for Science and Engineering*, vol. 49, no. 3, p. 3285–3312, 2024.

[25] C. Stathakopoulou and C. Cachin, "Threshold signatures for blockchain systems," *Swiss Federal Institute of Technology*, vol. 30, p. 1, 2017.
[26] L. Wei. "Lizz: Bitcoin Compatible, Conditional and Flexible Multi-Party Payment Channel,". Accessed: 2024 - 08 - 05.
[27] K. Xiao, J. Li, Y. He, X. Wang, and C. Wang, "A secure multi-party payment channel on-chain and off-chain supervisable scheme," *Future Generation Computer Systems*, vol. 154, p. 330–343, 2024.
[28] Y. Ye, Z. Ren, X. Luo, J. Zhang, and W. Wu, "Garou: An efficient and secure off-blockchain multi-party payment hub," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, p. 4450–4461, 2021.
[29] Y. Zhang and N. Jan, "Research on multiparty payment technology based on blockchain and smart contract mechanism," *Journal of Mathematics*, vol. 2022, p. 1–14, 2022.

Biography

Leiyang Wei is currently pursuing the M.E. degree in Zhejiang University of Technology, Hangzhou, China. His research interests include blockchain, cryptography and network security.

Zhicheng Xu received the B.E. degree from the College of Computer and Technology, Zhejiang University of Technology, Zhejiang, China, in 2017. He is currently pursuing the Ph.D. degree in Computer Science with the College of Computer and Technology, Zhejiang University of Technology, Zhejiang, China. His research interests include blockchain, collaborative learning, and artificial intelligence.

Yiqiao Song is currently pursuing the M.E. degree in Zhejiang University of Technology, Hangzhou, China. Her research interests include blockchain, cryptography and information security.

Hongbing Cheng (Member IEEE) received the Ph.D. degree from the Nanjing University of Posts and Telecommunications. He is currently a Professor in college of computer Science, Zhejiang University of Technology and has published numerous research papers in high-quality international journals and conferences. Prof. Cheng served as invited editor of several international journals in some international conferences; and has been invited to give keynote speeches and chair committees, reviewed papers for many international journals and conferences. His research interests include blockchain, cryptography, privacy preserving and information security, computer communications and cloud computing security.