A Novel Off-Chain Transaction Framework for Channel Cooperation

Wei-Jun Gao, Cheng-Yin Jiao, and Yu-Ying Zhang
(Corresponding author: Cheng-Yin Jiao)

School of Computer and communication, Lanzhou University of Technology
No. 36, Peng-Jia-Ping Road, Lanzhou 730050, China
Email: 19993596498@163.com

(Received July 6, 2024; Revised and Accepted Mar. 7, 2025; First Online Mar. 14, 2025)

Abstract

Payment channels, which allow two parties to conduct micropayments without involving the blockchain, have become a mainstream method for enhancing the scalability of decentralized ledgers like Bitcoin and Ethereum. The expansion of these payment channels has led to the creation of payment channel networks, enabling users to make payments by utilizing existing channels as relay channels for a certain transit fee. As the number of transactions in the network increases, the issue of transaction failures due to insufficient funds at one end of the channel becomes more pronounced. This paper designs an off-chain payment protocol that addresses channel imbalance and insufficient funds at channel ends. Through channel cooperation, the protocol enables nodes at both ends of the channel to proportionally allocate available funds, thereby increasing the available funds on the optimal path, shortening the payment path, and improving the success rate of transactions. Our protocol ensures the atomicity of payments and can prove that participants' funds are not lost even in the presence of malicious parties. In addition to formal modeling and structural security, we conducted simulation experiments on the channel cooperation protocol and the multi-path split payment routing algorithm. Experimental evaluations indicate that compared to the multi-path split payment scheme, the channel cooperation protocol improves the payment channel network's success rate by approximately 16.1% and 13.25% for different fund transactions and routing hops. Additionally, the higher the participation rate of channel cooperation participants results the shorter the payment path, and the safer the payment process.

Keywords: Blockchain; Channel Cooperation; Channel Imbalance; Off-chain Payment Channel Network

1 Introduction

With the rapid development of blockchain technology, there is a growing demand for secure and efficient digital asset transaction methods. One of the most pressing technical barriers of blockchain is its limited transaction throughput. The widespread use of Bitcoin [1, 21] and Ethereum [6] has led to long delays and high fees [30]. Blockchain payment channels, as a solution to scalability, offer a new possibility for achieving instant, low-cost peer-to-peer payments. Blockchain payment channels establish temporary off-chain payment channels to transfer transactions off-chain, reducing on-chain storage pressure [13, 19]. This allows users to conduct secure and private fund transfers without needing to record every transaction on the blockchain.

Off-chain payment networks [8, 9] involve two users locking their funds on the blockchain, conducting transactions off-chain, and interacting with the blockchain only during the channel creation and closure [10]. At any time, parties can close the payment channel by submitting the funds to the blockchain, returning their tokens to their accounts. To further enhance scalability, multiple channels can be connected to form a Payment Channel Network (PCN) [15], where payments can be routed through multiple channels to reach the receiver [23,26]. For each pending payment, some funds must be reserved in the channel as collateral until the payment is completed. Therefore, intermediate nodes may charge a processing fee [5]. The longer the path, the more funds need to be reserved in the channel, resulting in higher processing fees for users, lower security, and reduced success rates.

In a large channel network payment process, it is crucial to find one or more routes with sufficient capacity on each link between the sender and the receiver. [25, 27] Although the sender is aware of the channels in the network and their initial funding, the sender is unaware of the current channel capacity changes that may have occurred due to previous payments (the sender only knows the current capacity of the channels they are involved in) [24]. Therefore, the sender can only guess which routes will be successful. Particularly when the transaction amount is high, it is likely that the channel capacity on the selected path will be insufficient, leading to payment failure.

We have designed a novel channel routing algorithm that aligns optimally with the requirements of our model. This algorithm effectively combines the shortest path first algorithm with the maximum flow algorithm, initially using the shortest path algorithm to identify low-latency paths, and then optimizing bandwidth and capacity through the maximum flow algorithm. This dual strategy ensures that while seeking the shortest path, we also adhere to channel capacity constraints, thereby avoiding transaction failures due to insufficient funds.

This approach significantly enhances the success rate of high-value transactions within the payment network. The algorithm's flexibility allows it to quickly adapt to various conditions in complex network environments, thus improving overall transaction efficiency. By shortening transaction paths and reducing the number of hops required, we not only accelerate transaction response times but also bolster transaction security.

Furthermore, the algorithm can dynamically adjust, enabling real-time updates to path selection based on changes in network status. This characteristic ensures that the algorithm maintains robust performance even in high-frequency trading scenarios. Through this innovative approach, we provide strong support for the stability and reliability of payment networks, contributing to the advancement of the entire financial ecosystem. Additionally, the scalability of the protocol means it does not require all nodes in the payment network to deploy the protocol. When the single flow fund capacity is sufficient for payment, transactions can proceed normally. If the fund capacity is insufficient, the protocol can be signed with adjacent nodes to complete the payment, ensuring unlinkability for payments and enhancing overall security. The modular description of the protocol allows for its instantiation using various routing algorithms, making it applicable whether the sender opts for the shortest path or the lowest cost path.

2 Background

2.1 Payment Channels

Payment channels are a promising technology that can enhance the scalability and efficiency of the main blockchain. Payment channels enable two mutually untrusting parties to transact without submitting each transaction record to the blockchain. Both parties broadcast the initial transaction along with the initial funds to the blockchain to open a payment channel. The fund capacity of the opened payment channel equals the total initial funds of both parties. The core of the payment channel protocol is that both parties reach a consensus on the latest fund distribution and prevent fraudulent rollback.

Virtual channels [18] are established on top of two payment channels running on the ledger or through already established virtual channels, allowing nodes connected via virtual channels to transact. Unlike payment channels, the creation of virtual channels does not require on-chain transactions. [22, 29] The advantage of virtual payment channels is that intermediaries do not need to confirm each transaction. However, the establishment of virtual channels requires intermediate nodes to lock coins. Similarly, the more recursion occurs, the greater the amount of locked funds. Perun [7] and GSCN [9] are typical examples of virtual channels.

Payment hubs expand payment channels by directly connecting multiple nodes. It allows users connected to the same payment hub to freely engage in off-chain payment activities. TumbleBit [14] and NOCUST [17] are specific implementations of different payment hubs. For payment hubs, they are suitable for blockchain networks with small, high-density transactions, but the downside is that they cannot reuse deposits in the payment channels, leading to fund fragmentation.

2.2 Payment Channel Networks

Payment channel networks [4] are an innovative network structure for cryptocurrency transactions that utilize existing channels as intermediaries rather than creating new payment channels or conducting on-chain transactions. The most important aspect is the enforced atomicity of transactions: either all intermediate channel capacities in the path are updated, or none are changed. Payment channel networks achieve secure fund transfers through Hash Time-Locked Contracts (HTLC) [3] technology. HTLC is a conditional contract whose terms are enforced by the blockchain system, eliminating the need to rely on the trust of any single participant in the network. This contract allows funds to be locked under certain conditions and automatically released when the conditions are met, or returned to the contract owner when the contract times out.

In a payment channel network, when a payment needs to be made, the receiver generates a random secret value R and sends its hash value (denoted as y, where y=H(R)) to all intermediate nodes on the payment path. Each intermediate node uses the hash value y to create an HTLC, locking a portion of its tokens, with the amount equal to the payment amount plus a certain routing fee.Once the payment is initiated, the receiver subsequently reveals the secret value R to complete the transaction. This action releases the locked tokens at each intermediate node, thereby ensuring a secure transfer of funds.

The Lightning Network and the Raiden Network are currently the most popular payment channel networks, utilizing HTLC-based interactive mechanisms to process payments. Recent studies have also proposed several improvements, such as Sprites [19], which batch payment transactions and instantly split them into multiple smaller payments, using a multipath transmission protocol to achieve high-throughput routing solutions in PCNs. Batching payments allows Spider to complete large transactions over low-capacity payment channels over time, increasing the probability of payment success even without sufficient funds in a single outgoing channel. RE-

VIVE [16] and Circle [2] proposed channel rebalancing, a strategy that adjusts the distribution of funds by transferring funds from channels in one direction to channels in the other direction. A reasonable fund distribution will increase the success rate of large payments. However, these two schemes may lead to multiple small payments for a single transaction or balancing fund transfers occupying multiple channels, causing network congestion [28] when transaction volumes are high. This method also assumes mutual trust among users on the loop path, making it difficult to find a loop path without causing imbalances in other channels. Shaduf [20] proposes adjusting funds in adjacent channels as a solution to this problem. This method does not require a loop path but only involves the node, adjacent nodes, and channels. However, this method also requires occupying funds from other channels, which can cause imbalances in those channels.

3 Scheme Design

3.1 Design Ideas

In our research, we adopted a channel-to-channel collaborative communication method to ensure the security of off-chain transactions. This communication method allows users to conduct unlimited transactions between channels after a single on-chain binding. Specifically, we designed a mechanism that treats the channel as a whole to complete transactions, regardless of the fund distribution at both ends of the channel. As long as the channel capacity is greater than the transaction amount, the transaction can be completed, thus improving the success rate of large payments.

In this scheme, the funds allocated to users in each intermediary channel are used proportionally, and the intermediary fees are also determined by the proportion of funds used. This design not only ensures the effectiveness of the transaction but also ensures its fairness. Additionally, the protocol ensures that the honest party does not lose funds, thereby enhancing user trust [6].

In each transfer of funds between a channel and an adjacent channel, three nodes are involved, with each node acting as a sender, intermediary, and receiver in the transaction. This method of transaction provides unlinkability, effectively preventing intermediaries from colluding or stealing funds from the channels, thereby thus ensuring the security of users' funds.

Furthermore, the scheme does not rely on all nodes in the network, but only involves users' own channels and adjacent users, thereby effectively resisting uncooperative participants. Any three connected and mutually trusted users can complete large transactions below the channel capacity using this method, achieving effective fund flow and redistribution, thereby enhancing the efficiency of the entire payment channel network.

3.2 Detailed Design

3.2.1 Model Definition

Current large-scale payment solutions focus on dividing large payment transactions into several small payment transactions, completing payments through multi-path methods. In contrast, channel collaboration schemes increase the maximum amount of funds that can be completed on the shortest path. In short, after users bind on-chain once, they can perform bidirectional transactions on adjacent channels multiple times, completing transactions for amounts less than the channel capacity.

After users establish a one-time on-chain binding, they can directly transfer tokens between two bound channels without limitation on the number of transactions, facilitating bidirectional payments.

First, we define the basic structure of channel binding as illustrated in Figure 1.

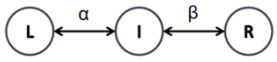

Figure 1: Diagram of the channel structure

Where L, I, R are user nodes, α, β are channels between users, θ is the total amount of funds sent, and *value* is the channel capacity. When during a transaction event it passes through a node where the funds on one side of the channel are insufficient to complete the transaction, but the channel capacity with the previous node exceeds the total amount of funds sent, this node can act as an intermediary and initiate a channel cooperation request to the adjacent nodes on both ends.

Each time the intermediary node I initiates a channel cooperation request to the adjacent channels α and β, the request includes the allocation of funds sent from α_L end and α_I end, as well as from β_I end and β_L end. When participating nodes L and R respond with signed participation information, the channel cooperation agreement is reached, the fund allocation process is illustrated in Figure 2 below. The three-party nodes involved in this process must reach consensus within $O(\Delta)$ time; if any party fails to respond, the channel cooperation agreement cannot take effect.

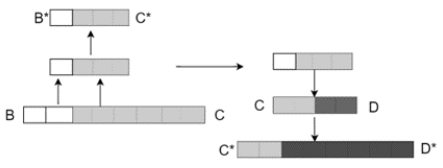

Figure 2: Channel allocation of funds

When channel α transfers θ to channel β, nodes I and R sign and confirm completion within the specified time

$O(\Delta)$, marking the transaction as successful. The channel capacity is updated, the channel cooperation version is incremented, and it is broadcasted.

If any user participating in the transaction requests to terminate the channel cooperation or raises a dispute before all fund transfers are completed, the transaction is canceled. The channel reverts to its original capacity, and the channel cooperation version is restored.

The channel cooperation contract discourages any user from broadcasting transactions other than the most recent channel cooperation version. If a user incorrectly broadcasts an old channel version, all their funds are penalized and delivered to the other party. If participating users agree on the latest channel version, either user can broadcast a transaction revocation. After a waiting period, funds can be reused.

3.2.2 Algorithm Definition

Algorithm 1 constructs a network topology G based on the model, calculating a path from one node to another with the minimum edge (channel) weight that still satisfies 70% of the channel capacity for the transaction amount. This algorithm employs a greedy strategy, first declaring an array to store the shortest distances between nodes and a set of nodes for which the shortest path has been found, while setting the lengths of unreachable paths to infinity. When a new transaction is added, the algorithm checks if there is a reachable path and if the length of this path is the shortest one that satisfies the transaction amount. If so, the array values are updated accordingly.

algorithm 1 SWM

initialize graph G with nodes and edges
initialize capacity function $C(e)$ for each edge e in G
initialize flow function $F(e) = 0$ for each edge e in G
initialize source node s and sink node t
while there exists an augmenting path P from s to t in residual graph **do**
    find the minimum residual capacity C_r along P
    if $C_r \geq 0.7 \times C(e)$ for all edges e in P **then**
        for each edge e in P **do**
            augment flow $F(e)$ by C_r
            reduce capacity $C(e)$ by C_r
            increase reverse flow $F_{rev}(e)$ by C_r
        end for
    end if
end while
initialize shortest path algorithm
find shortest path SP from s to t based on residual capacities
ensure that for each edge e in SP, $F(e) \leq 0.7 \times C(e)$
return SP and corresponding flow values

Algorithm 2 takes the transaction amount, sender node, and receiver node as inputs and outputs the result of either a successful or failed transaction. The specific steps of Algorithm 2 are as follows:

algorithm 2 Cooperate

Input: Transaction amount θ, Sender node S, Receiver node R
Output: "Transaction Success" or "Transaction Failure"
if funds at S are insufficient to complete transaction θ **then**
    Identify previous node N_{prev} and next node N_{next}
    Identify channel capacities $C(\alpha)$ between N_{prev} and S, and $C(\beta)$ between S and N_{next}
    if $C(\alpha) \geq \theta$ and $C(\beta) \geq \theta$ **then**
        Mark S as an intermediary node
        Calculate fund allocation for α_L and α_S
        Calculate fund allocation for β_S and β_R
        Initiate channel collaboration request to adjacent channels α and β
        Request includes fund allocations for α_L and α_S, and for β_S and β_R
        if participants respond with signed participation information **then**
            Channel collaboration agreement is reached
            Execute transaction
            Return "Transaction Success"
        else
            Channel collaboration fails, terminate transaction
            Return "Transaction Failure"
        end if
    else
        Channel capacity insufficient, terminate transaction
        Return "Transaction Failure"
    end if
else
    Execute transaction directly
    Return "Transaction Success"
end if

- **Input**: Transaction amount θ, Sender node S, Receiver node R.

1) Handling Insufficient Funds If the funds at the sender node S are insufficient to complete the transaction amount θ, then identify the previous node N_{prev} and the next node N_{next}, as well as the channel capacities $C(\alpha)$ and $C(\beta)$ between them and S. If both $C(\alpha)$ and $C(\beta)$ are greater than or equal to θ, then mark S as an intermediary node, calculate the fund allocation for channels α and β, and initiate a channel cooperation request to the adjacent channels.

2) Channel Cooperation Request The cooperation request includes the fund allocation for α_L and α_S, as well as β_S and β_R.

   If the participating nodes respond with signed confirmation, then a channel cooperation agreement is reached, the transaction is executed, and "Transaction Success" is returned.

   Otherwise, if the cooperation fails, the transaction is terminated, and "Transaction Failure" is returned.

3) Direct Transaction Execution If the funds at the

sender node S are sufficient to complete the transaction, the transaction is executed directly, and "Transaction Success" is returned.

- **Output**: "Transaction Success" or "Transaction Failure".

3.3 Model Implementation

This paper formalizes the model using the universal composability(UC) framework [12] [11], employing a global setup where the model consists of a fixed set of participants P $=\{P_1, ..., P_n\}$. The underlying blockchain is viewed as a global ideal function ζ representing maximum blockchain latency. All participants receive inputs from the environment Z and send them accordingly.

Channels:The payment channel δ is defined by attribute $(\delta.id, \delta.users, \delta.balance, \delta.fund, \delta.bind, \delta.allot)$.In terms of attributes, δ.id $\in \{0,1\}^*$ is the channel identifier, $\delta.users = \{\delta.lift, \delta.right\}$ denotes the set of channel users, δ.balance represents the channel balance, δ.fund denotes the channel capacity, δ.bind records all bindings of this channel, and δ.allot indicates the remaining capacity after binding to other channels.

Bindings: The binding is defined by the attribute tuple (Φ.id, Φ.users, Φ.channels, Φ.shift, Φ.reserve). Regarding attributes, Φ.id $\in \{0,1\}^*$ is the binding identifier, $\Phi.users = (\Phi.lift, \Phi.originator, \Phi.right)$ represents the set of users, where Φ.originator acts as the intermediary. $\Phi.channels = (\Phi.\alpha, \Phi.\beta)$ denotes the Cooperation of two bound channels, where $\Phi.\alpha$ consists of users Φ..lift and Φ.originator, and $\Phi.\beta$ consists of Φ.originator and Φ.right. Φ.shift represents the total shifted coins between channels with $\Phi.shift(\Phi.a) + \Phi.shift(\Phi.\beta) = 0$. Φ.reserve indicates the upper limit for each channel's capacity transfer.

Communication: We denote the event "sending message m to P in round r" as " $m \xrightarrow{r} p$ ", and the event "receiving message m from P in round r" as " $m \xleftarrow{r} p$ ", where P can be a transaction party or a smart contract. Specifically, when P is a set of participants, the above notation represents sending a message to each member of the set or receiving a message from each member of the set.

Communication Network: This paper assumes a synchronous communication network where all parties are aware of the current time, measured in rounds. Message delivery between communication parties takes one round, meaning a message sent in round r is received by the recipient at the beginning of round r+1. Additionally, communication is authenticated by each user, allowing the receiver to verify the message source. Attackers can view message contents but cannot modify or delete messages.

Ledger: This paper formalizes the ledger as a global function ζ relative to the blockchain ledger. The tuple $(x_1, ..., x_n)$ recorded in the ledger represents the balance x_i of participant P_i,accessible to all users. Its functionality is depicted as shown in Figure 3.

Contract Functionality: The model denotes a contract

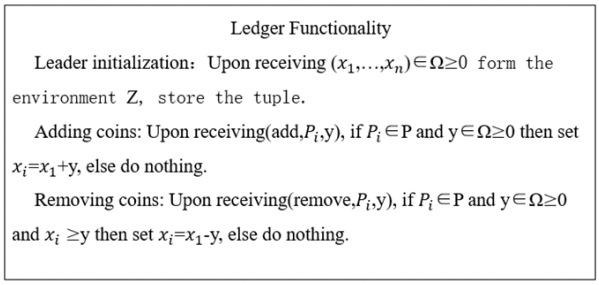

Figure 3: Ledger functionality

using f, where each contract is identified by its corresponding channel. For instance, $f(\beta)$ represents the contract of channel β. f accesses the contract ledger.

The channel Cooperation process is triggered by messages from users, including process names and references to channel or binding identifiers. Assuming the following messages involve channels β and α: channel identifiers are denoted as id, the payment initiator as P, other users as Q. The requested payment amount and fund transfer are denoted as θ and $\tilde{\theta}$, respectively. The ledger is ζ, and the list of channels is Γ.

Security Objective: Ensuring the security of funds for honest users, meaning honest users will not lose funds during the transaction process. We analyze the expected implementation of contracts in each stage as follows.

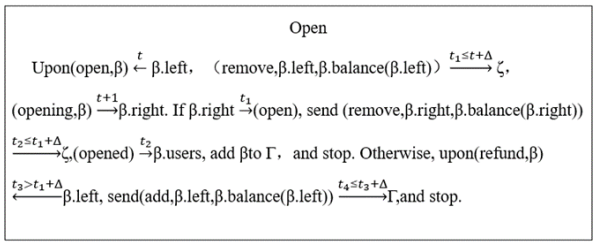

Figure 4: Channel open process

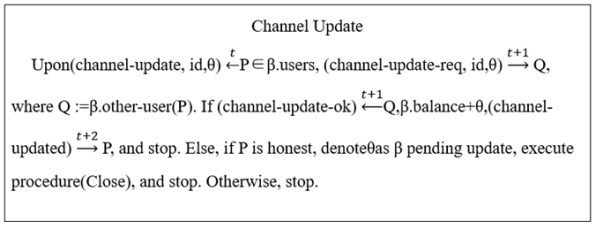

Figure 5: Channel update process

Channel Opening and Channel Update Contracts: The channel opening is only considered agreed upon when the initiator Originator applies, and users Light and Right approve within O(Δ) time. Similarly, the channel status

update is completed within $O(\Delta)$ time when all participating users agree. When all three users act honestly, it takes 2 rounds to confirm the result.The process is shown in Figure 4 and Figure 5.

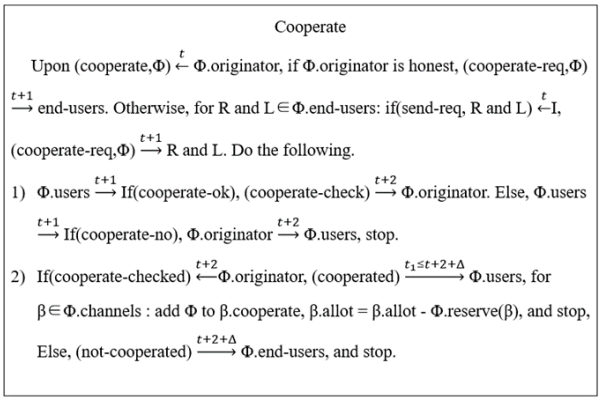

Figure 6: Channel cooperate process

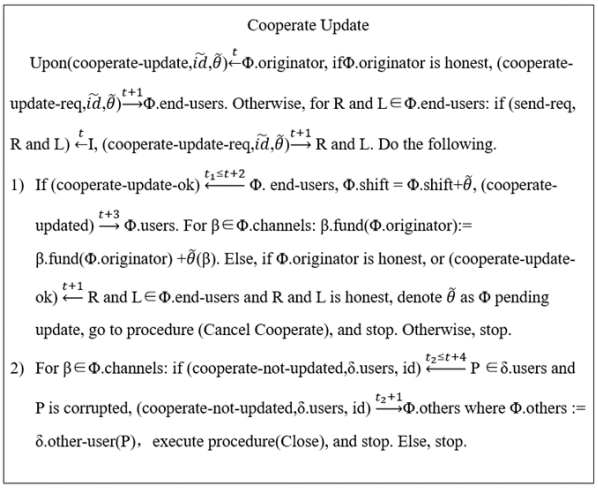

Figure 7: Channel cooperation update process

Channel Opening Contract: The process of opening a payment channel is considered finalized only when the initiating party, referred to as the Originator, submits an opening request, and the other two participants, Light and Right, give their approval within a bounded time frame denoted as $O(\Delta)$. This timeframe $O(\Delta)$ refers to a pre-specified delay tolerance in the system, ensuring that all participants have a fair opportunity to review and approve the channel opening. The use of multi-party signatures ensures that the agreement is verifiable on-chain, mitigating risks such as double-spending or collusion.

Channel Update Contract: For any subsequent update to the channel state, such as adjusting balances or modifying contract terms, all participants must consent to the changes within $O(\Delta)$. A consensus mechanism is employed to facilitate agreement, where the update is only finalized once all parties sign off on the revised state. This consensus ensures that no single party can unilaterally alter the channel's status, preserving the integrity and fairness of off-chain interactions.

In scenarios where all participants behave honestly and without delay, both the channel opening and channel update processes can be completed in 2 communication rounds. This minimal communication overhead is a significant advantage of off-chain networks, allowing for rapid finality in payment verification. The entire sequence of actions, including the signing, approval, and channel confirmation, is illustrated in Figure 6 and Figure 7, which demonstrate the step-by-step interaction of all parties involved.

Channel Dissolution Cooperation Contract: In a payment channel network, when transactions within a channel are no longer needed or users wish to exit the channel, a channel dissolution cooperation contract can be proposed to terminate the channel.

Contract Initiation: Any user from the two cooperating channels can propose a channel dissolution cooperation contract. The initiator sends a dissolution request to other participants, explaining the reasons for termination and the method of fund settlement. The initiator plays a critical role in the process as it triggers the subsequent negotiation and confirmation stages.

Consensus Mechanism: Once the channel dissolution cooperation contract is initiated, all participating users must reach a consensus within $3O(\Delta)$ time for the channel to be officially dissolved. Here, $3O(\Delta)$ represents three times the system's tolerated delay time. During this time window, all participants must provide explicit consent or denial. To ensure security and fairness, a multi-signature mechanism is often introduced during the dissolution process, allowing every participant's decision to be recorded and verified.

Dissolution Process: Dissolving a channel involves not only closing the channel but also reallocating any remaining funds within the channel. The contract must clearly outline how the funds are to be distributed according to each party's share, ensuring that all users' funds are returned to their accounts after the channel is closed. This redistribution process typically involves multi-signature validation and on-chain verification to ensure transparency and security.

Handling Failure to Reach Consensus: If consensus is not reached within the $3O(\Delta)$ time frame, the dissolution request will be considered a failure, and the channel will remain open. At this point, the initiator may choose to reissue the dissolution request or continue conducting transactions within the channel. To avoid malicious behavior or prolonged deadlock, some protocols may introduce arbitration mechanisms or incentives to encourage participants to reach consensus within a reasonable time frame.The process is shown in Figure 8.

Closure Process:The user initiating the closure sends a closure request to all participating nodes within the channel. The request specifies whether there are any pending

Cancel Cooperate

Upon(cancel-cooperate,$\tilde{id}$,$\tilde{\theta}$)$\xleftarrow{t}$P$\in\Phi$.users, or be invoked by other procedures, in round $t_1 \le t + \Delta$, if Φ pending update and $\tilde{\theta}$ is confirmed, Φ.shift = Φ.shift+$\tilde{\theta}$, and for $\beta\in\Phi$.channels: β.fund(Φ.originator):= β.fund(Φ.originator) +$\tilde{\theta}$(β).(cancel-cooperate) $\xleftarrow{t_2\le t_1+2\Delta}$ Φ.users. For $\beta\in\Phi$.channels, do the following.

1) β.fund := β.fund + Φ.shift(β), β.allot = β.allot +Φ.shift(β) + Φ.reserve(β), and remove $\Phi$from β.cooperate.
2) If (cooperate-not-updated,δ.users, id) $\xleftarrow{t_2\le t+4}$ P$\in\delta$.users and P is corrupted, send(cooperate-not-updated,δ.users, id) $\xrightarrow{t_3+1}\Phi$.others where Φ.others := δ.other-user(P), execute procedure (Close) and stop. Else, stop.

Figure 8: Cancel cooperate process

Close

Upon (close, id) $\xleftarrow{t}$ P$\in\beta$.users, or be invoked by other procedures (let t be the current round), in round $t_1 \le t + \Delta$, if βpending update$\theta$is confirmed, β.balance = β.balance+θ. Do the following.

1) If β.cooperate=none,(add, remove,β.left,β.balance(β.left)) $\xrightarrow{t_2\le t_1+2\Delta}\zeta$, (add,$\beta$.right,$\beta$.balance($\beta$.right)) $\xrightarrow{t_2}\zeta$,(closed) $\xrightarrow{t_2}\beta$.users, remove $\beta$from Γ, and stop.
2) Otherwise, for $\Phi\in\beta$.cooperate: execute procedure (Cancel Cooperate). Let t_3 be the current time. (add, remove,β.left,β.balance(β.left)) $\xrightarrow{t_4\le t_3+2\Delta}\zeta$, (add,$\beta$.right,$\beta$.balance($\beta$.right)) $\xrightarrow{t_4}\zeta$,(closed) $\xrightarrow{t_4}\beta$.users, remove $\beta$from Γ, and stop.

Figure 9: Channel close process

transactions or collaboration contracts that need to be resolved.

For the closure to proceed, all channel participants must agree on the final balance and confirm the closure within the 3Δ time frame.If consensus is reached, the channel is closed, and the final state is recorded on the blockchain. This step ensures that the integrity of funds is preserved, and no participant suffers any loss due to abrupt termination.

Dissolution of Collaboration Contracts:If collaboration contracts are present, the system will first trigger the dissolution process for these contracts. This includes notifying all linked channels and reallocating funds to ensure that the dependencies between channels are fully severed. Collaboration dissolution must also occur within the same 3Δ time frame, requiring agreement from all parties involved in the external contracts.

Final Confirmation and Blockchain Settlement: Once all dependencies are resolved, the final balance is confirmed and signed by all users. The channel closure is then recorded on the blockchain as a final, irreversible state change. This prevents any further transactions or claims on the closed channel.The process is shown in Figure 9.

4 Experiments and Analysis of Results

4.1 Experimental Design

This experiment collected information on 1,000 nodes and channels of the Lightning Network on December 27, 2023. The frequency distribution of channel capacities is shown in Figure 10. As illustrated in the figure, most channel capacities in the Lightning Network range between 0 BTC and 3 BTC, with only a few having larger capacities.

n the simulated transaction process, a random value N_i is generated based on the capacity of each channel. This random value ranges between 0 and $CN_{i,j}$, and the sum of the values at both ends of the channel CN_i and

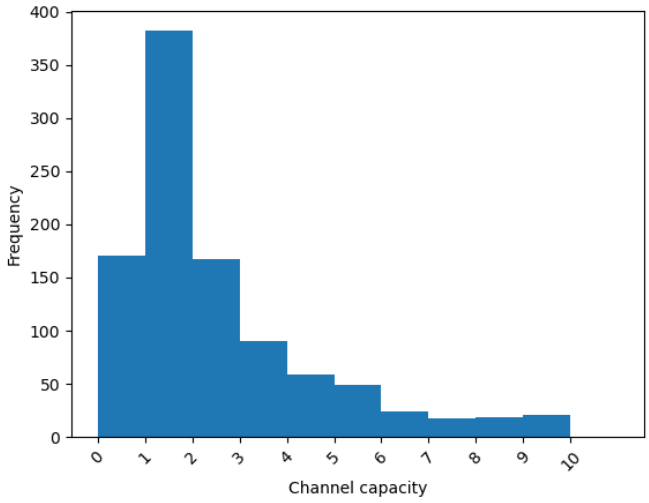

Figure 10: Channel capacity frequency

CN_i equals the channel capacity $CN_{i,j}$, i.e (Formula 1):

$$CN_i + CN_j = CN_{i,j} \tag{1}$$

Among the 100 channels, most exhibit imbalance issues, with a few channels showing extreme imbalance where the funds at one end are zero. According to the imbalance Formula (2):

$$\frac{|CN_i - CN_j|}{CN_{i,j}} = b_{i,j}^{im} \tag{2}$$

The skewness of funds at both ends of the selected 100 channels was analyzed, as shown in Figure 11. The green line in the figure represents a perfectly balanced state, with an imbalance degree of zero, indicating the ideal optimal balance. The orange line in the figure represents the actual balance state of the channels. As can be seen, the vast majority of channels are in a non-ideal state. The

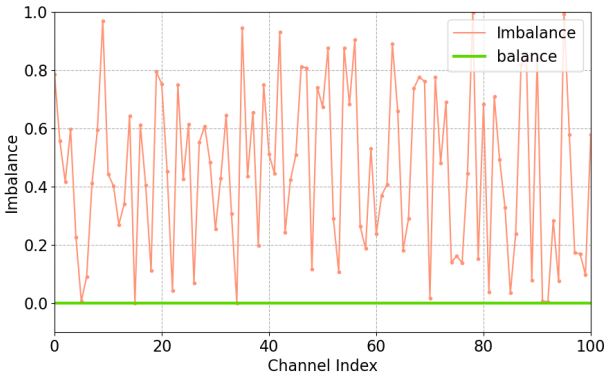

Figure 11: Channel imbalance

larger the imbalance degree, the more one-sided the fund flow at both ends of the channel, making it more likely to experience fund shortages in future transactions.

4.2 Experimental Results and Analysis

Ethereum uses gas to measure the amount of computational resources required to perform certain operations. Each instruction executed by the Ethereum Virtual Machine (EVM) consumes a certain amount of gas. There is no fixed price for converting gas to Ether. Because the sender of the transaction specifies the gas price, it affects the miners' willingness to process the transaction. A lower gas price will result in a longer waiting time for the transaction to be mined. We selected an average gas price of approximately 20 Gwei (or 2×10^{-8} Ether) as the standard price for the experiment. At the time of preparing this paper, the exchange rate of Ether to USD was 1:2920.32. That is to say,1 gas $= 2\times10^{-8}$ Ether $= 5.841\times10^{-5}$ USD.

Our evaluation uses the following criteria: the number of on-chain transactions, the execution cost measured in gas, Ether, and USD, the number of off-chain messages, and the number of signatures. The number of signatures dominates in terms of message length and computational complexity. Table 1 below shows the execution cost of each operation under these metrics.

We then simulate payment networks of different sizes to evaluate the effectiveness of the channel cooperation method in improving the success rate of off-chain payments and shortening the average payment path length. First, we establish underlying payment networks of various sizes, randomly opening a certain number of payment channels in the network. Then, we extract transactions from the Lightning Network and simulate the transaction process on the payment network.

In a small-scale payment network, the ratio of nodes to channels determines the network's density. The higher the density, the shorter the average path length between nodes, resulting in more potential paths for completing transactions. We chose a ratio of 4.0 for the simulation of the payment network. Another important system parameter in the experiment is the participation rate in channel cooperation, which indicates how many participants will engage in channel cooperation. Table 2 shows the simulation results. Here, PN stands for "Payment Network", PH stands for "Payment Hub", FW stands for the Floyd-Warshall shortest path algorithm, SM stands for the SpeedyMurmurs routing algorithm, and PC stands for the "Channel Cooperation" routing algorithm. Λ represents the improvement compared to the payment network.

From the simulation experiments, it can be observed that the channel cooperation algorithm effectively shortens the transaction path length. Moreover, its effectiveness in path shortening improves with an increasing rate of participant usage.

To simulate real dynamic transaction data and test the model's success rate, this study uses static data of 5000 nodes and channels, conducting 1000 transaction simulations each time, with random selection of senders and receivers. These simulations are compared against the performance of multi-path splitting payments in off-chain payment channels.

Initial fund simulation in off-chain payment channels: Due to the invisibility of initial balance distribution within each channel, we designed a method to simulate payment scenarios by randomly distributing the channel capacity between two channel users.

Off-chain payment scenario simulation: Due to the invisibility of transaction frequency and amounts, we need to set the payer and payee for each transaction pair and determine the payment amounts. Since payment channels are primarily used for high-frequency, small-amount transactions, we excluded excessively high transaction amounts. Instead, we randomly sampled transaction parties within the network and conducted high-frequency tests over a short period to better simulate real off-chain network transactions.

We designed evaluation methods for the channel cooperation and SpeedyMurmurs algorithms. For the SpeedyMurmurs algorithm, we adopted the optimal fund partitioning ratio recommended within this simulation scenario. For the channel cooperation algorithm, users need to transfer funds within bound channels. To ensure the cooperation of the other two users in the bound channel during the transaction, we need to initiate the binding for each user. The binding amount is the portion of a channel's funds available for use by one user. To prevent the bound channel from becoming underfunded and unable to complete other transactions after a binding transaction, we limit the binding fund to 70% of the total channel capacity, with this amount distributed according to the initial fund allocation between the two channel users. If the cooperative channel is an intermediary channel, the resulting transit fees are distributed proportionally to the fund allocation ratio.

We selected the transaction volume range of 0-3 BTC, where transactions are more frequent in the payment

Table 1: Operating cost

Operation	On-chain txs	gas	ether	USD	Off-chain msgs	sigs
Open	2	173147	0.0034	9.94	0	2
In-channel transfer	0	0	0	0	2	2
Cooperate	1	154723	0.0030	8.77	0	2
Cross-channel transfer	0	0	0	0	17	17
Withdraw	2	97749	0.0019	5.56	0	2
Close	2	148413	0.0029	8.48	0	2

Table 2: Route length comparison

Number of nodes	PN-FW	PH-FW	PC-FW	Λ	PN-SM	CH-SM	PC-SM	Λ
200	2.77	2.74	2.52	9.0%	4.01	3.99	3.65	9.1%
400	3.10	3.03	2.76	11.1%	4.71	4.58	4.03	14.5%
600	3.30	3.19	2.91	11.8%	5.06	4.76	4.32	14.5%
800	3.44	3.32	2.98	13.4%	5.33	4.92	4.27	20.0%
1000	3.56	3.40	3.03	15.5%	5.53	5.13	4.20	24.2%

channel network. Assuming that the number of transits for all payment paths is 3, a comparison of the multi-path transaction success rates between our scheme and the Spider scheme is shown in Figure 12.

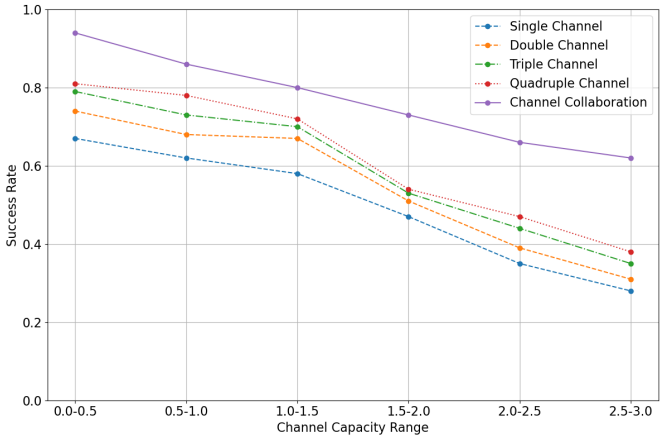

Figure 12: Comparison of transaction success rates 1

In the simulation experiments, as the transaction amount increases, the success rate of various path transaction algorithms shows a significant downward trend. This is due to the decrease in the number of channels that can satisfy the transaction amount and the increase in the degree of channel imbalance. However, in the experimental graph, it is evident that multipath partitioned payments can improve the transaction success rate. The channel cooperation-based algorithm proposed in this paper shows a more significant improvement in transaction success rate, with smaller fluctuations as the transaction amount increases, improving the success rate by nearly 16.1% compared to the optimal multipath algorithm.

We set the transaction range to 0-1 BTC to explore the multipath transaction success rate comparison between our scheme and the Spider scheme under varying numbers of payment path hops. the success rate is increased by 13.25% on average compared with multi-path split payment.The results are shown in Figure 13.

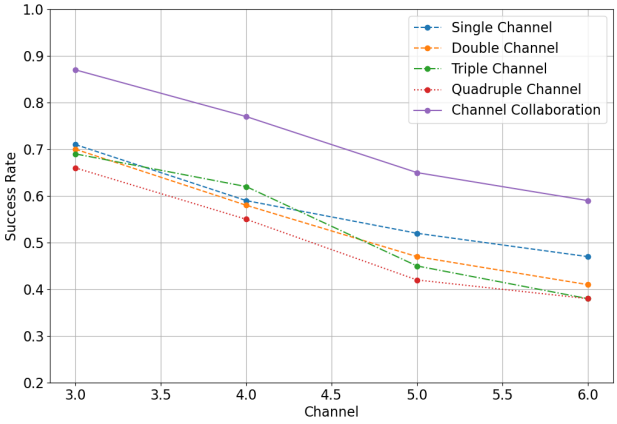

Figure 13: Comparison of transaction success rates 2

The utilization of a single channel is constrained by its inherent fund capacity, which lacks the ability for dynamic adjustment. When funds are insufficient, transactions may fail, resulting in wasted resources. In contrast, collaboration with adjacent channels enables dynamic allocation of funds and maximizes their utilization, significantly enhancing overall transaction success rates and efficiency. As illustrated in Figure 14, channel collab-

oration leads to a substantial improvement in fund utilization, reduces the number of payment routing hops, and lowers processing fees, thereby enhancing the overall economic viability of transactions—particularly for large-scale transactions.

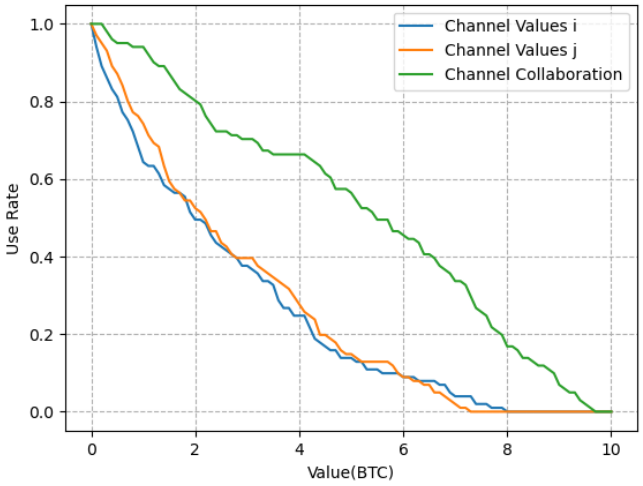

Figure 14: Channel fund utilization rate

The channel cooperation model distinguishes itself from other channel protocols primarily through its flexibility and dynamic adaptability. Unlike traditional channel protocols, the channel cooperation model facilitates collaboration among adjacent channels, thereby enhancing the maximum capacity for fund flows. This cooperative mechanism not only improves the success rate of transactions but also reduces the number of payment routing hops, ultimately lowering transaction costs.

Enhanced Capacity Utilization: The channel cooperation model allows multiple channels to work in tandem, optimizing fund utilization. This capability ensures successful transactions even in scenarios of insufficient funds.

Increased Success Rate: By dynamically adjusting fund allocations, the model can identify more optimal transaction paths, significantly improving the success rate of large transactions within the payment network.

Flexibility: The model can adapt to changes in network conditions in real-time, addressing varying transaction volumes and channel states, which is particularly beneficial for high-frequency trading scenarios.

Security: The channel cooperation model offers a degree of unlinkability; even in cases of collusion among intermediate nodes, the model prevents the linking of different parts of the same transaction, thereby enhancing transaction privacy and security.

Modular Design: This model features a modular protocol design that allows for integration with various routing algorithms, making it applicable across diverse scenarios.

5 Conclusions

The off-chain transaction method based on channel cooperation proposes a transaction protocol initiated by intermediate nodes with cooperation between adjacent channels. This protocol ensures atomicity, correctness, and security in the transaction process. Our experimental evaluation demonstrates the advantages of channel cooperation in transaction success rates and shortening payment paths. The main contributions of this paper are as follows:

- Based on channels, we extended the amount of funds that can be paid through a channel, increasing the success rate of large payments.
- Effectively improved payment efficiency in imbalanced networks, reducing transaction fees and path hops, allowing more transactions to be completed on the optimal path.
- The contract ensures the security of users' transaction funds, enhancing transaction reliability.

Off-chain payment methods through channel collaboration not only enhance the success rate of payments but also reduce the number of hops in payment routing. This approach shows promise for application in networks such as the Lightning Network and the Raiden Network, facilitating larger transactions and advancing the payment sector. However, this method lacks dynamic analysis when faced with high-frequency transactions, potentially leading to transaction delays and channel imbalances.

Future work will focus on researching off-chain payment channel networks, aiming to design a payment method suitable for high-frequency transactions that dynamically adjusts the channel network. Ensuring user and fund security is also crucial. By employing information encryption and control mechanisms, we can enhance the security of off-chain payments. Additionally, integrating smart contracts could automate and secure the process, while decentralized oracles might provide real-time data to dynamically balance channels, ensuring the network remains efficient and reliable even under high transaction loads.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (No.61762059). The authors are also particularly grateful to the reviewers for their suggestions.

References

[1] T. Alam, "A survey on the use of blockchain for the internet of things," *International Journal of Electronics and Information Engineering*, vol. 13, no. 3, pp. 119–130, 2021.

[2] Z. Avarikioti, K. Pietrzak, I. Salem, S. Schmid, S. Tiwari, and M. Yeo, "Hide & seek: Privacy-preserving rebalancing on payment channel networks," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 358–373.

[3] C. Burchert, C. Decker, and R. Wattenhofer, "Scalable funding of bitcoin micropayment channel networks," *Royal Society open science*, vol. 5, no. 8, p. 180089, 2018.

[4] C. Decker and R. Wattenhofer, "A fast and scalable payment network with bitcoin duplex micropayment channels," in *Stabilization, Safety, and Security of Distributed Systems: 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings 17*. Springer, 2015, pp. 3–18.

[5] J. Dong, G. Xu, C. Ma, J. Liu, and U. G. O. Cliff, "Blockchain-based certificate-free cross-domain authentication mechanism for industrial internet," *IEEE Internet of Things Journal*, 2023.

[6] J. Dong, G. Xu, C. Ma, J. Liu, and U. G. O. Cliff, "Certificate-free cross-domain fine-grained access control mechanism for industrial internet," *Human-Centric Computing And Information Sciences*, vol. 14, 2024.

[7] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment channels over cryptographic currencies." *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 635, 2017.

[8] S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski, "Perun: Virtual payment hubs over cryptocurrencies," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 106–123.

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

[10] C. Egger, P. Moreno-Sanchez, and M. Maffei, "Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 801–815.

[11] P. Frauenthaler, M. Sigwart, C. Spanring, M. Sober, and S. Schulte, "Eth relay: A cost-efficient relay for ethereum-based blockchains," in *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2020, pp. 204–213.

[12] Z. Ge, Y. Zhang, Y. Long, and D. Gu, "Shaduf: Non-cycle payment channel rebalancing." in *NDSS*, 2022.

[13] N. Hamian, M. Bayat, M. R. Alaghband, Z. Hatefi, and S. M. Pournaghi, "Blockchain-based user re-enrollment for biometric authentication systems," *International Journal of Electronics and Information Engineering*, vol. 14, no. 01, pp. 18–38, 2022.

[14] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted bitcoin-compatible anonymous payment hub," in *Network and distributed system security symposium*, 2017.

[15] P. Hoenisch and I. Weber, "Aodv–based routing for payment channel networks," in *Blockchain–ICBC 2018: First International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 1*. Springer, 2018, pp. 107–124.

[16] R. Khalil and A. Gervais, "Revive: Rebalancing off-blockchain payment networks," in *Proceedings of the 2017 acm sigsac conference on computer and communications security*, 2017, pp. 439–453.

[17] R. Khalil, A. Gervais, and G. Felley, "Nocust-a non-custodial 2nd-layer financial intermediary." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 642, 2018.

[18] P. McCorry, M. Möser, S. F. Shahandasti, and F. Hao, "Towards bitcoin payment networks," in *Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I 21*. Springer, 2016, pp. 57–76.

[19] 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*. Springer, 2019, pp. 508–526.

[20] P. Moreno-Sanchez, A. Blue, D. V. Le, S. Noether, B. Goodell, and A. Kate, "Dlsag: non-interactive refund transactions for interoperable payment channels in monero," in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 2020, pp. 325–345.

[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[22] M. Peck, *Understanding blockchain technology: abstracting the blockchain*. IEEE, 2018.

[23] S. Roos, P. Moreno-Sanchez, A. Kate, and I. Goldberg, "Settling payments fast and private: Efficient decentralized routing for path-based transactions," *arXiv preprint arXiv:1709.05748*, 2017.

[24] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 777–796.

[25] A. Tomar and S. Tripathi, "Blockchain-assisted authentication and key agreement scheme for fog-based smart grid," *Cluster Computing*, vol. 25, no. 1, pp. 451–468, 2022.

[26] P. Wang, H. Xu, X. Jin, and T. Wang, "Flash: efficient dynamic routing for offchain networks," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 370–381.

[27] Z. Wang, L. Yang, Q. Wang, D. Liu, Z. Xu, and S. Liu, "Artchain: Blockchain-enabled platform for

art marketplace," in *2019 IEEE international conference on blockchain (blockchain)*. IEEE, 2019, pp. 447–454.

[28] H. Xue, Q. Huang, and Y. Bao, "Epa-route: Routing payment channel network with high success rate and low payment fees," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 227–237.

[29] K. Yang, D. Li, Q. Guo, H. Wang, D. Bai, and X. Pan, "Research on deep forgery data identification and traceability technology based on blockchain," in *2022 IEEE 2nd International Conference on Data Science and Computer Application (ICDSCA)*. IEEE, 2022, pp. 230–234.

[30] Y. Zhang, D. Yang, and G. Xue, "Cheapay: An optimal algorithm for fee minimization in blockchain-based payment channel networks," in *ICC 2019-2019 ieee international conference on communications (icc)*. IEEE, 2019, pp. 1–6.

Biography

Wei-Jun Gao received his Bachelor of Science degree from Lanzhou University in 1997 and his Master of Science degree from Chinese Academy of Sciences in 2000.He has been engaged in teaching and scientific research at Lanzhou University of Technology since 2000. His research interests include software engineering, distributed computing and cloud computing, big data processing, and graphics and image processing.

Cheng-Yin Jiao received her B.S. degree in Majored in Computer Science and Technology from Bohai University in 2022, and is now studying for her M.S. degree in School of Computer and Communication at Lanzhou University of Technology. Her main research interests are network and information security, and blockchain.

Yu-Ying Zhang received her B.S. degree in Majored in Computer Science and Technology from Shandong Modern University in 2022, and is now studying for her M.S. degree in School of Computer and Communication at Lanzhou University of Technology. Her main research interests are Natural language processing.