Enabling Cross-chain Transactions: A Decentralized Cryptocurrency Exchange Protocol

Hangyu Tian, Kaiping Xue, *Senior Member, IEEE,* Xinyi Luo, Shaohua Li, Jie Xu, Jianqing Liu, Jun Zhao, David S.L. Wei, *Senior Member, IEEE*

***Abstract*—Inspired by Bitcoin, many different kinds of cryptocurrencies based on blockchain technology have turned up on the market. Due to the special structure of the blockchain, it has been deemed impossible to directly trade between traditional currencies and cryptocurrencies or between different types of cryptocurrencies. Generally, trading between different currencies is conducted through a centralized third-party platform. However, it has the problem of a single point of failure, which is vulnerable to attacks and thus affects the security of the transactions. In this paper, we propose a distributed cryptocurrency trading scheme to solve the problem of centralized exchanges, which can achieve secure trading between different types of cryptocurrencies. Our scheme is implemented with smart contracts on an Ethereum blockchain and deployed on an Ethereum test network. In addition to implementing transactions between individual users, our scheme also allows transactions among multiple users. The experimental result proves that the cost of our scheme is acceptable.**

***Index Terms*—Blockchain, Cryptocurrency, Exchange, Ethereum, Smart contracts.**

I. Introduction

Bitcoin [1] is currently the most popular cryptocurrency which has been running well until today since it was deployed in 2009. Unlike traditional currencies, Bitcoin does not rely on a central issuer for management, but rather runs on a P2P (Peer to Peer) network, which means no central authority can fully control Bitcoin. Its transaction data is not stored in a central database, but is written in a distributed hash chain named blockchian. In recent years, inspired by Bitcoin, various blockchain-based cryptocurrencies have emerged, such as Litecoin [2] and Ethereum [3]. To help users manage different kinds of cryptocurrencies, a centralized exchange based on a trusted third party is commonly used in some recent works, e.g., [4–8].

H. Tian is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, Anhui 230027, China.

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

S. Li is with the Department of Computer Science, ETH Zurich, Zurich 8092, Switzerland.

J. Xu is with the Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong.

J. Liu is with the Department of Electrical and Computer Engineering, University of Alabama in Huntsville, Huntsville, AL 35899, USA.

J. Zhao is with the School of Computer Science and Engineering, Nanyang Technological University, 639798, Singapore.

D. Wei is with the Computer and Information Science Department, Fordham University, NY 10458, USA.

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

On the one hand, different kinds of cryptocurrencies are deployed in different blockchains by leveraging different blockchain technologies. Transactions between the same currency can be made directly on the network though a blockchain clients. However, cryptocurrencies deployed on different blockchains cannot be traded directly with each other. Furthermore, if a user wants to buy some cryptocurrency using the traditional currency or convert his/her cryptocurrency into cash, the user can only exchange them with a third party. In such cases, an exchange based on a trusted third party is helpful, which can act as an intermediary to help users trade between different types of cryptocurrencies.

On the other hand, the owner of a cryptocurrency typically manages his/her assets through an address bounding to a pair of key which is always stored in his/her private device. In this situation, once the equipment fails, loses or even is attacked by a malicious malware against cryptocurrency, the user suffers property loss. So a central organization is needed to help users manage their properties. A centralized exchange plays this role which not only serves as a platform for cryptocurrency trading but also provides a place for users to store their property. Just like a bank, a centralized exchange provides users with the convenience of managing funds and conducting transactions. There are now more than 4,000 centralized exchanges in the world, such as South Korea's Bithumb [9], which trades over $1 billion of cryptocurrencies a day.

Although centralized exchanges provide user convenience in trading, it also brings about some security risks. Once users keep their properties in a centralized exchange platform, it means that the exchange platform is the "Archilles Heel" of the system which could result in malicious use of users' properties and transaction information. Furthermore, As a central institution [10, 11], there will always be a single point of failure. The best way to solve this problem is to adopt a distributed cryptocurrency exchange scheme, which is also in line with the idea of decentralization of cryptocurrency. Smart contracts [12, 13] are codes that can be deployed and executed on a blockchain. With smart contracts, we are able to implement a variety of decentralized applications. Currently, Ethereum is the largest and most popular blockchain platform supporting the deployment of smart contracts. Users can send their smart contract codes to the Ethereum network through transactions, which can then be verified by miners and added to the blockchain. Any smart contract code saved in the blockchain can be invoked by the users who meet certain conditions. In this paper, we consider using smart contracts based on Ethereum to implement a decentralized cross-cryptocurrency exchange scheme which can verify different types of cryptocurrency transactions sent by different users. We not only resolve transactions between individual users, but also consider the situation where trading is done among multiple users.

In this paper, we make the following key contributions:

- We propose a decentralized cross-cryptocurrency exchange scheme based on smart contracts, in which, by using randomly selected users as intermediaries, transactions between any two types of cryptocurrencies can be realized in single-user and multi-user scenarios. In the latter, multiple users can initiate multiple transfers in a short time period, and the established contract can automatically collect these transfers and complete these transactions simultaneously.
- In our scheme, through Proof of Work and Proof of Deposit, untrusted willing users are randomly selected to organize a validation committee. Furthermore, by using smart contracts to collect judgment results from the members in validation committee, our scheme can validate transactions between any two different types of cryptocurrencies via the Ethereum network. Further analysis shows that the functionality and transaction atomicity of our scheme can be guaranteed by the protocol design of the selection validation committee.
- We implement and deploy our proposed scheme on an Ethereum test network, and evaluate the running costs of each part of the contract on our local machine. Our scheme optimizes the execution performance of the same type of cryptocurrency among multiple users. Experimental results show that the local operation cost of our scheme is only affected by the number of participants and is independent of the number of transactions per user. Our cross-cryptocurrency transaction scheme ensures that deployment costs and execution costs are within the acceptable range to users.

The rest of our paper is organized as follows. Section II introduces a series of work related to our scheme, and then in Section III technical preliminaries are presented. After we introduce our system and security model in Section IV, Section V details our cross-cryptocurrency transaction scheme. Section VI provides a security analysis of our scheme. Section VII shows the detailed deployment of our scheme and the related performance analysis. Finally, we have a summary of our work in Section VIII.

II. Related Work

In this section, we introduce the existing works on decentralized cryptocurrency trading schemes which are designed to solve the single point of failure problem in centralized exchanges.

The Metronome project [14] proposes a cryptocurrency called MTN that can be traded across different blockchains. While a user destroys the token on the source chain, he/she receives a proof of exit receipt that can be used on the target blockchain. However, Metronome can only be implemented in blockchains that support smart contracts, and cryptocurrencies that do not support smart contracts thus cannot be exchanged.

KyberNetwork [15] is a highly liquid chain protocol that provides instant trading and redemption services for digital assets and cryptocurrencies which is currently deployed on Ethereum. In KyberNetwork network, users store their tokens in the repository by interacting with smart contracts. The users can also access other types of cryptocurrency through the repository which can be created by the contract or by a third-party agency. The funds in the repository created by the third-party agency are sourced from the fund provider, and a reserve manager maintains the repository, determines the exchange rate, and feeds the ratio back to KyberNetwork. The addition and deletion of reserve entities in the network are the responsibility of the KyberNetwork operator. KyberNetwork relies on the block chain relay technology (such as BTCRelay [16]) to achieve cross-chain confirmation, so there are defects in the support of multiple currencies.

ERC-20 [17] is a standard interface for tokens on the Ethereum blockchain. Some work, e.g., [18–21], try to resolve transactions between ERC-20 tokens on Ethereum and bitcoin. But these works only serve ERC-20 tokens on Ethereum, which has great limitations. Republic [22] is a decentralized dark pool project between cryptocurrency pairs across different blockchains. Dark pool provides a hidden order book where financial assets and instruments are traded and matched by an engine built on a multi-party computation protocol. Dogethereum [23] is a peer to peer internet currency that runs smart contracts and enables the exchange of dogecoins for an equivalent worth of ethereum tokens and vice versa. However, Republic and Dogethereum only provide exchanges between a limited variety of tokens and Ethereum. However, in our scheme, we use smart contracts on Ethereum to verify other types of cryptocurrencies through a validation committee selected from intermediary nodes, so our scheme is able to support transactions between any types of cryptocurrencies.

Both Cosmos [24] and Polkadot [25] are working to solve the interoperability of blockchains, using the consensus algorithm of Tendermint [26]. The blockchain in Cosmos network applies a central radiation model: at the center is the "Hub", which manages a number of independent blockchains called "Zones", and tracks the status of all zones. Each zone is obligated to continuously produce itself through reporting the new block back to the Hub. Similarly, there are four basic roles in maintaining Polkadot network: collators, fisherman, nominator, and validator. They construct a series of child blockchains which enforce state via merkle proofs. But both the two works only support the blockchain network that is compatible with them, and both need to have new tokens issued by the new network, which increases the transaction burden.

XCLAIM [27] is a generic framework for cryptocurrency to achieve untrusted and efficient cross-chain switching. The disadvantage is that the currency on the issuing blockchain requires a contract that supports certain functions. So a limited-capacity scripting language such as Bitcoin does not support this operation. Tesseract [28] describes how to mark existing cryptocurrencies with a trusted execution environment (TEE) to enable cross-chain transactions, while TEEs suffer from their own security concerns such as rollback [29] and side-channel attacks [30].

Atomic Cross-Chain Exchange (ACCS) [31, 32] based on hashed timelocks or signature locks [33–36] enables secure cross-chain switching, but there is limitation in practicality. Such an ACCS scheme is interactive, relying on a synchrony assumption. In this way, long waiting time is often incurred during transmission. Different from their ideas, our protocol relies on weak/partial synchronization assumption, and can achieve good performance in the case of multiple transaction participants.

III. Preliminaries

A. Consensus Algorithms in Blockchain

In a blockchain system, the technology used to ensure the consistency of distributed nodes is known as consensus algorithm. Currently commonly used consensus algorithms are mainly divided into two types. The first is the proof-based consensus, such as PoW [1], PoS [37], DPoS [38] , GHOST [39] and so on, which require the user to submit a solution to a difficult problem to the network. Only the node verified by the network can publish the block, which is also the commonly used consensus in cryptocurrency. The other is the traditional distributed network algorithm to solve the Byzantine problem, including PBFT [40, 41], RAFT [42], and so on. Traditional distributed network algorithms are often used in the design of the permissioned blockchain.

B. Gas in Smart Contract

The concept of using gas limits the length of smart contracts deployed on Ethereum, which also prevents malicious users from deliberately deploying contracts that contain an infinite loop, causing Ethereum to crash. Gas is leveraged to measure the cost of each step in the smart contract code. Each transaction is required to include a gas limit which is the maximum amount of gas consumed by the transaction, and the amount of Ether that the user pays for each unit of gas. Miners have the right to choose which transaction to package first. The more transaction fees are payed, the more likely a miner is to package your transaction, and the faster the transaction will be confirmed. Gas limit is the transaction fee that the user need to pay at most, which limits the ability of a smart contract to execute without restriction. In this paper, we deploy our contract to test network based on Ethereum, measured the gas needed for each transaction, and calculate the true deployment and operational costs based on the current exchange rate.

IV. System Model, Security Model, and Design Goals

A. System Model

There are four main components in our system model, which are described in detail as follows:

- ***Payer.*** A payer is a user who wants to transfer cryptocurrency to another user.
- ***Payee.*** A payee is a user who needs transfer-in, but the type of cryptocurrency he/she needs is not available on the payer's side.
- ***Intermediary.*** An intermediary is a user who has an Ethereum account and an account of another type of cryptocurrency, and has enough property in these accounts to be used for conducting transactions. Intermediaries connect the payer and the payee through smart contracts. Intermediaries need to join the validation committee to participate in the transaction validation process, which will be introduced in Section V-D.
- ***Blockchain.*** Each execution of our scheme involves three blockchains, in which two blockchains support cryptocurrencies used respectively by the payer and the payee. The third blockchain is the Ethereum blockchain, on which our smart contracts are designed and deployed.

The notations are summarized in Table I.

TABLE I
Symbol definition.

Symbol	Description
$\mathcal{A}$	The payer of a transaction
$\mathcal{B}$	The payee of a transaction
$\mathcal{C}_1$	The first intermediary of a transaction
$\mathcal{C}_2$	The second intermediary of a transaction
$coin_1$	Cryptocurrency owned by the payer
$coin_2$	Cryptocurrency required by the payee

Provided that we deploy smart contracts on Ethereum, we further assume that all users participating in our scheme need to have their own Ethereum accounts and only a small amount of ethers are required to pay for the fees of calling contract. Meanwhile, in our scheme, it's not required for payer/payee to have sufficient ethers to support their transactions straightly. For a user who only needs to participate in the trade in our scheme, he/she can join the blockchain network as a SPV (Simplified Payment Verification) node [1]. We believe that it is not so easy to find an intermediary who can support both two types of cryptocurrencies required for a transaction across the two cryptocurrencies. Even if such a intermediary exists, we can treat it as two intermediaries we mentioned in our scheme. Therefore, we use two suitable intermediaries, e.g., $\mathcal{C}_1$ and $\mathcal{C}_2$, to realize a cryptocurrency transaction between $\mathcal{A}$ and $\mathcal{B}$ respectively. In addition, each intermediary has the ability to verify different kinds of cryptocurrency transactions through wallet, blockchain browser and some other tools. Since the intermediary is required to verify transactions, it can be a full node or SPV node.

In our scheme, we adopt the time assumption of weak/partial synchronization, which means messages are guaranteed to be delivered after a certain time bound. It is difficult to distinguish between normal nodes and malicious nodes that transmit overtime in a completely asynchronous network. As a result, in our schme, a timer will be set in the smart contract to prevent malicious nodes from refusing to provide verification results, so as to distinguish between normal nodes and malicious nodes that have transmit overtime.

B. Security Model

Malicious users in our system may set up a large number of accounts to act as payers, payees, or intermediaries and send a large number of junk transactions to undermine our scheme. All malicious users may collude to gain extra benefits by defrauding the contract. Their possible malicious behaviors are shown below.

- Malicious payers or intermediaries may send spoofing messages to a contract without making transfers or make double spending.
- Malicious payees or intermediaries can defraud the contract after receiving transfers to earn additional compensation fees.
- Malicious nodes participating in the validation committee (described in Section V-D) may release wrong information to disrupt the consensus process.

Unlike malicious nodes, trusted nodes will comply with the terms of our protocol to enforce their behavior in order to achieve cross-cryptocurrency transactions or earn transaction fees through our scheme. Since the validation committee is elected by proof-of-work, we assume that for the nodes who want to join the validation committee, the combined hash power of the malicious nodes should be less than $1/4$ of the total hash power at any time.

Otherwise, it will introduce selfish mining attacks. This attack will make malicious nodes more likely to generate more blocks that exceed their own computing power, thereby allowing them to obtain more seats than the percentage of computing power they possess in the validation committee.

C. Design Goals

We aim to design a decentralized secure transfer scheme between different types of cryptocurrencies. Specifically, our design goals include the following:

- ***Decentralized and non pepudiation***: There will not be any centralized third party in our scheme. Any party involved in the agreement cannot deny their own behavior. Any user's operation on the smart contract will be permanently recorded on the blockchain.
- ***Portability***: In our cross-blockchain cryptocurrency transactions scheme, we need to support transactions between multiple types of cryptocurrencies. In other words, our transaction scheme is not restricted by the types of cryptocurrencies.
- ***Fairness***: Even if any step in our scheme fails, our scheme still guarantees that the system is in normal operation and will not harm the interests of any honest users.
- ***Stability***: Our protocol needs to be able to withstand common attacks in blockchain networks under security assumptions.
- ***Scalability***: In actual transaction scenarios, there are not only transactions between two single users, but also transactions among multiple users. In our scheme, we need to support transactions among multiple users, and the performance needs to be acceptable.
- ***Atomicity***: In our cross-blockchain transactions scheme, if all parties conform to the protocol, then all swaps take place. Then, no conforming party ends up worse off, and no coalition has an incentive to deviate from the protocol [43].
- ***Safety and liveness***: For a consensus protocol, safety means that honest (non-Byzantine) nodes agree on the same value, and liveness means that the transaction will eventually abort or commit.

V. Cross-cryptocurrency Transaction Scheme

In this section, we first explain how our scheme is applied to two users who trade with different cryptocurrencies. Then, we design a secure verification method to verify that the transaction is completed. Finally, we extend the scheme to the scenario where multiple users trade through different cryptocurrencies.

A. Overview

In this section, we give an overview of our scheme to show how we enable transactions across different kinds of cryptocurrencies. Our scheme consists of three phases:

- ***Contract deployment phase.*** This phase includes the deployment process of the three types of contracts involved in our scheme.
- ***Transaction phase.*** The payer and payee of the transaction complete the cross-cryptocurrency transaction with the participation of intermediaries in this phase. In our scheme, this phase is divided into two situations including single-user transaction and multi-user transaction.
- ***Transaction validation phase.*** In order to verify the transaction results of other types of cryptocurrency in the Ethereum smart contract, we verify the transaction by selecting a validation committee in this phase.

For the convenience of illustration, we take Bitcoin and Litecoin for examples. In fact, our scheme supports transactions between any kind of cryptocurrencies. As illustrated in Fig. 1, to achieve a cross-cryptocurrency transaction through smart contracts, we need two intermediaries $\mathcal{C}_1$ and $\mathcal{C}_2$ who can be anyone in the network.

In contract deployment phase, a large number of contracts are deployed in the network. Then the transaction takes place in transaction phase. Intermediaries need to support transactions through Bitcoin and Litecoin respectively. Firstly, $\mathcal{A}$ transfers x litecoins to $\mathcal{C}_1$. After receiving the transfer, $\mathcal{C}_1$ transfers the equivalent ethers to $\mathcal{C}_2$, and then $\mathcal{C}_2$ transfers y bitcoins to $\mathcal{B}$. This way, we use the transfer of ether as a bridge to achieve the transfer of different cryptocurrencies between two users. The incentive for the user to participate in our contract as an intermediary is that he/she can get transaction fees through this process.

It should be noticed that in addition to the transfer of ether in Ethereum, there is also transfer of bitcoin and litecoin. Although the transactions of these two currencies cannot be directly validated through the Ethereum smart contract, we find a solution by selecting a group of users as a validation committee to provide verification results. In transaction validation phase, The contract will integrate the judgment results of the committee and draw the final conclusion. As concluded in work [44], cross-chain communication is impossible without a trusted third party, and the validation

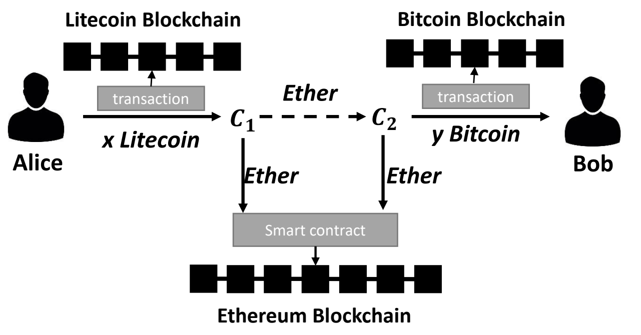


Fig. 1. An overview of the scheme.

committee composed of distributed nodes plays such a role in our scheme.

In addition to the single-user transaction scenario mentioned above, we also consider the multi-user transaction scenario. In this case, there are multiple payers who want to transfer cryptocurrencies to one or many payees. Our contract could combine payers who need to transfer the same kind of cryptocurrency with a payee who needs a different kind of cryptocurrency. The details of our full proposal will be described in Section V-C1 to V-C2.

B. Contract Deployment Phase

In our scheme, there are three types of smart contracts written in Ethereum's Solidity language.

- ***Intermediary contract.*** This smart contract primarily controls the behavior of the intermediary, and provides an interface for obtaining information about the intermediary node.
- ***Transaction contract.*** By this contract the payer and payee of the transaction complete the cross-cryptocurrency transaction with the participation of intermediaries. And this contract provides a call interface for input for verification results.
- ***Committee contract.*** This contract is used for validation committee elections, including PoW puzzle verification and controlling the size of the validation committee, and provides an interface to verify whether the node is a member of the validation committee.

The main functions of our contracts are listed as follows. We denote intermediary contract as IC and transaction contract as TC and committee contract as CC.

TABLE II
SMART CONTRACT FUNCTION.

Contract steps	Function
IC.Register	Intermediary provides registration information
IC.Update	Intermediary update information
CC.Verify_PoW	Verify wether the user is eligible to join the verification group
TC.Prepare	Information preparation for trading users
TC.Deposit	User submits deposit
TC.validation	User's transaction verification

In contract deployment phase, these three types of contracts are deployed in large numbers across the network. Users and intermediaries can freely choose contracts that do not exceed the gas limit for transactions. Once the transaction demands in the network exceed the number of contracts, additional transaction contracts and committee contracts can also be further deployed by any intermediaries. The intermediary transaction information that users can query in the intermediary contract includes the transaction contract address of the contract deployed by the intermediaries. An intermediary may deploy multiple contracts. Any intermediary can join other contracts as one of the two intermediaries required for the transaction. Transactions through different contracts occur in parallel, and hence the more transaction contracts there are in the network, the higher the transaction efficiency will be under the fixed number of users.

C. Transaction Phase

1) Single-user Transaction Phase: In transaction phase, we first introduce the single-user transaction scheme. The single-user transaction scheme represents that both parties involved in the transaction are individuals. This scheme also includes a user's currency exchange with himself/herself (e.g., Alice converts her bitcoin to ether). For clarity, we assume that the exchange rate between $coin_1$ and ether and the exchange rate between $coin_2$ and ether are both "1". In the real scenario, the problem of non-"1" exchange rate only needs to multiply the rate with the corresponding magnification. The main procedures of the single-user transaction scheme are shown in Algorithm 1.

Algorithm 1: Framework of one-to-one transaction scheme

```
Input: The account addresses of both parties and the
       intermediary; Transaction amount $x$; Exchange
       rate; Time threshold $T$;
if Both intermediaries $\mathcal{C}_1$ and $\mathcal{C}_2$ have deposited Ether then
    Payer $\mathcal{A}$ transfers $xcoin_1$ to intermediary $\mathcal{C}_1$;
    if Transfer confirmed successful then
        Intermediary $\mathcal{C}_2$ transfers $xcoin_2$ to payee $\mathcal{B}$;
        if Transfer confirmed successful then
            Return the deposit of intermediary $\mathcal{C}_2$ to himself;
        else if Timeout or confirmation fails then
            Transfer the deposit of intermediary $\mathcal{C}_2$ to $\mathcal{B}$;
        end
        Transfer the deposit of intermediary $\mathcal{C}_1$ to $\mathcal{C}_2$;
    else if Timeout or confirmation fails then
        The deposit of intermediary $\mathcal{C}_1$ and $\mathcal{C}_2$ are returned;
    end
else if $\mathcal{C}_1$ or $\mathcal{C}_2$ did not provide a deposit within time $T$ then
    Return the deposit to $\mathcal{C}_1$ or $\mathcal{C}_2$ then terminate;
end
```

Before the transaction begins, the intermediaries issue their own supported cryptocurrency types and corresponding exchange rates to the contract. The parties of the transaction including $\mathcal{A}$ and $\mathcal{B}$ select the appropriate intermediaries $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, and publish information about their transactions to the contract, such as cryptocurrency types, transaction amount, and account addresses. Then, the intermediaries $\mathcal{C}_1$ and $\mathcal{C}_2$ receive information about the transaction of both sides and notify contract about the users they select. Then, they need to send a certain amount of deposit to the contract within time T. If any intermediary fails to provide deposit within time T, the contract will return the deposit of the other intermediary and stop the protocol. After receiving the message that payer $\mathcal{A}$'s transfer to $\mathcal{C}_1$ has succeeded, intermediary $\mathcal{C}_2$ transfers $x\ coin_2$ to payee $\mathcal{B}$. Otherwise, if $\mathcal{A}$'s transfer is judged to be overtime or verified as failed, the contract returns the deposit of $\mathcal{C}_1$ and $\mathcal{C}_2$ and terminates the transaction. If the final transfer transaction is successful, the deposit of $\mathcal{C}_2$ will be returned to himself. Otherwise if $\mathcal{C}_2$'s transfer is judged to be overtime or verified as failed, the deposit saved in the contract by intermediary $\mathcal{C}_2$ will be sent to $\mathcal{B}$. In the end, intermediary $\mathcal{C}_2$ will receive the deposit of $x\ ethers$ from $\mathcal{C}_1$.

2) Multi-user Transaction Phase: In many cases, there may be multiple users trading simultaneously in a short time period. Our smart contract is designed to support multiple users to participate in cryptocurrency transfer. In multi-user transaction phase, multiple users can individually select their own trading partners. The contract will combine these transaction information and aggregate the amount of cryptocurrency transferred to the same user to improve transaction efficiency.

The framework of multi-user trading scheme is shown in Algorithm 2. This scheme is mainly used in the scenario where a group of users trade with each other. Smart contract first collects information from all payers $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3......\mathcal{A}_n$ and works out the sum of the transaction amount. After intermediaries $\mathcal{C}_1$ and $\mathcal{C}_2$ receive this information, they are required to pick the appropriate users and submit the equivalent of ethers as deposit. If any intermediary fails to provide deposit within time T, the contract will return the deposit of the other intermediary and stop the protocol. Then, all the payers need to pay enough $coin_1$ to $\mathcal{C}_1$. Members of the validation committee need to verify these transfers through the methods described in Section V-D. The transfer information will also be recorded in the contract. After intermediary $\mathcal{C}_1$ receives the transfer from the payers successfully, the contract calculates the amount of successful transactions, modifies the amount received by $\mathcal{C}_1$, and then sends the message to $\mathcal{C}_2$. Otherwise, if any payer $\mathcal{A}_i$'s transfer is judged to be overtime or verified as failed, the contract will return the deposit equivalent to $amount_A[i]$ to $\mathcal{C}_1$ and $\mathcal{C}_2$. Intermediary $\mathcal{C}_2$ finally transfers $coin_2$ to each of the payees $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3......\mathcal{B}_n$ based on the received message. Otherwise, if the transfer to payee $\mathcal{B}_i$ is overtime or judged to fail, the contract will send the same amount of intermediary $\mathcal{C}_2$'s deposit to $\mathcal{B}_i$. In the end, the contract will also send the remaining t deposit of intermediary $\mathcal{C}_1$ to $\mathcal{C}_2$.

Algorithm 2: Framework of multi-user transaction scheme

```
Input: The account address of payers:
       $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3......\mathcal{A}_n$; payees: $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3......\mathcal{B}_n$
       and intermediaries:$\mathcal{C}_1$, $\mathcal{C}_2$; Total transaction
       amount:$t$; Amount to be sent by payers
       $\mathcal{A}_i$:$amount_A[i]$; Amount to be received by
       payees $\mathcal{B}_i$:$amount_B[i]$; Time threshold $T$;
Set each element in the array $amount_B$ to zero;
if Both intermediaries $\mathcal{C}_1$ and $\mathcal{C}_2$ have deposited Ether then
    All the payers $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3......\mathcal{A}_n$ transfer $coin1$ to
      intermediary $\mathcal{C}_1$;
    for $i = 1$ to $n$ do
        if Sender $\mathcal{A}_i$ transfers successfully then
            Record $\mathcal{A}_i$;
            Add the value $amount_A[i]$ to the array
              element $amount_B[j]$ corresponding to the
              payee $\mathcal{B}_j$ of $\mathcal{A}_i$;
        else if Timeout or confirmation fails then
            $t = t - amount_A[i]$; Transfer the deposit
              equivalent to $amount_A[i]$ to $\mathcal{C}_1$ and $\mathcal{C}_2$;
        end
    end
    Intermediary $\mathcal{C}_2$ transfers $coin2$ to the payees
      $\mathcal{B}_1, \mathcal{B}_2......\mathcal{B}_n$ separately;
    for $i = 1$ to $n$ do
        if The transfer to $\mathcal{B}_i$ is successful then
            Record $\mathcal{B}_i$;
            The deposit of $\mathcal{C}_2$ equivalent to
              $amount_B[i]$ will be returned to $\mathcal{C}_2$;
        else if Timeout or confirmation fails then
            Transfer the deposit of $\mathcal{C}_2$ equivalent to
              $amount_B[i]$ to $\mathcal{B}_i$;
        end
    end
    Transfer the remaining deposit of quantity $t$ of
      intermediary $\mathcal{C}_1$ to $\mathcal{C}_2$;
else if $\mathcal{C}_1$ or $\mathcal{C}_2$ did not provide a deposit within time $T$ then
    Return the deposit to $\mathcal{C}_1$ or $\mathcal{C}_2$ then terminate;
end
```

D. Transaction Validation Phase

The transfer between intermediary $\mathcal{C}_1$ and $\mathcal{A}$ and the transfer between intermediary $\mathcal{C}_2$ and $\mathcal{B}$ need to be confirmed. And the validation result needs to be sent to the contract for the next step. We cannot simply ask any user to participate in the transaction validation because malicious users may masquerade as both sides of the transfer. For example, it is possible for the payer to send a fraudulent message to the contract about a successful transfer without any actual transfer. Similarly, the payee may send a message that the transfer failed after receiving the currency. In transaction validation phase, we design a reliable method to remedy this problem by decentralizing the authority of transaction validation.

We consider that there are a large number of intermediaries in the system, but only two of them are needed for each transaction progress. Therefore, the rest of intermediary nodes could be used in our scheme as validators to verify the transaction. The intermediary earns fees by participating in the validation process. We also require each node to participate in a successful validation process at least once before becoming an intermediary. We assume that each intermediary has the ability to verify the transfer of different types of cryptocurrencies through wallet, block explorer, and many other tools. In general, different types of cryptocurrencies have their own confirmation policies. For example, in Bitcoin, recipients assume their transactions are secure with 6 blocks attached.

Algorithm 3: Validation committee selection

```
Input: Node account $acc$; The random number $nonce$; The last hash value $before$; Deposit value $value$;
Initialize: Difficulty of PoW: $target$; Minimum deposit: $deposit_{value}$; The current block hash: $hash_{now}$; Last consensus time: $time_{last}$; Mapping of accounts to committee members: $isMembers$; Current committee member address array: $committee_{member}[w]$; Number of committee members: $w$;
if $value \geq deposit_{value}$ && $before == hash_{now}$ then
    Record the current moment: $now$;
    Calculate the hash based on user input: $h = hash(hash_{now}||nonce||acc)$
    if $target \geq h$ && $isMembers[acc] == false$ then
        $target = target * \frac{10min}{now - time_{last}}$;
        $time_{last} = now$;
        $isMembers[acc] = true$;
        $hash_{now} = h$;
        if Size of array $committee_{member}$ is equal to $w$ then
            $isMembers[committee_{member}[0]] = false$
            Erase $committee_{member}[0]$
        end
        Add $acc$ to the last part of $committee_{member}$;
    else
        Termination;
    end
else
    Termination;
end
```

To prevent malicious users from registering multiple intermediary accounts to attack the validation process, we use the proof-of-work (PoW) algorithm to determine the set of intermediaries participating in the validation committee. The operation is completed by committee contract, and the algorithm is shown in Algorithm 3. Before joining the validation committee, inspired by proof-of-deposit, each intermediary who successfully solves the proof-of-work puzzle is required to pay some ethers as a security deposit. The proof-of-work puzzle requires the node to choose a random number based on the hash of the previous block and its own Ethereum account to make the hash result smaller than a target value. We set the difficulty of the proof-of-work puzzle such that each of the two intermediaries participating in committee takes about 10 minutes to solve. This way, all transactions generated within 10 minutes will be validated by the intermediary in this committee. We set the number of committee members to w, and require that the same account cannot join the committee twice. If an intermediary node is always honest, it will not only get back its deposit but also get rewards from the penalty of dishonest nodes and the transaction fee from the guarantee of the transaction. Thus, being an intermediary node in our system is monetarily profitable. The formation of the trade validation committee is shown in Fig. 2.

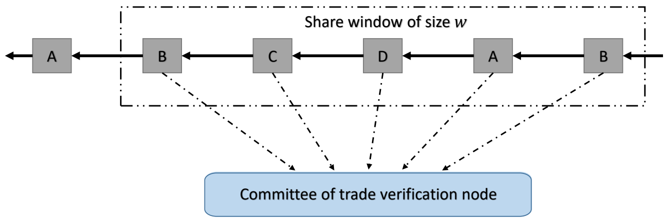


Fig. 2. Formation of a transaction validation committee.

After each transaction, every node in the validation committee needs to verify the transaction and sends its own validation results to the smart contract. The smart contract records the address of these nodes and counts the validation results. Then, the validation committee will classify nodes that give conflicting validation results. We use the majority rule to determine the final validation result. To this end, the nodes whose results conflict with the final result will be considered dishonest and thus are excluded from the validation committee. Their deposit will be equally distributed to the honest nodes. The correct result will serve as the basis for the next step of smart contracts. We have a quantitative limit of w for the size of the validation committee. By enforcing the committee size, all intermediaries' deposit are kept in the contract till any dishonest node is excluded from the validation committee. Similar to a timing policy, conforming a committee size allows the deposit of dishonest nodes to be transferred to the honest nodes once node exclusion is triggered.

VI. Security Analysis

In this section, we provide a security analysis of our scheme, and discuss how our scheme can mitigate or eliminate some well-known attacks against blockchain.

A. Decentralized and Non Repudiation

Our scheme does not require the participation of any trusted third party at all, but guarantees transaction security in a fully distributed manner through smart contracts and validation committees. The intermediaries and validation committees involved in the protocol are selected through a fully distributed security strategy.

In our scheme, apart from payer and payee, other entities involved only include the intermediary and the validation committee, neither of which is a centralized design. First of all, the two intermediaries required in our scheme are selected from many intermediaries. It is not easy for an attacker to predict the intermediaries that users will choose and attack in advance. And as long as there are enough intermediaries involved, our system will not fall back to a pseudo-centralized system. On the other hand, the validation committee is also selected among distributed nodes that have solved the PoW problem within a certain time period. The members of the committee will change over time, and the reliability of the members of the committee is guaranteed by PoW and the deposit. Therefore, our protocol neither relies on any centralized system nor falls back to a pseudo-centralized system.

And our scheme also realizes the non-repudiation after the transfer is executed. This is because once any participant sends or receives a transaction, the transaction will be stored in the blockchain through the smart contract and cannot be tampered with. And there is a validation committee to ensure the security of verification on the blockchain outside the smart contract, so neither participants could deny their own actions.

B. Portability

In our cross-blockchain cryptocurrency transactions scheme, we support transactions between multiple types of cryptocurrencies. For the transactions of cryptocurrencies other than Ethereum, we all use the method described in Section V-D, by selecting a validation committee to verify the transaction and provide the verification results to the contract. During the verification phase of the protocol, there might be malicious nodes in the validation committee who may provide incorrect verification results to disrupt the execution of our protocol. In our scheme, the validation committee is selected by proof-of-work and requires users to provide sufficient deposit. And in our scheme these two parts are indispensable.

On the one hand, PoW is used for dynamically choosing nodes to participate in the validation committee within a time period. If our scheme is designed to only rely on the absolute ability of participating in the validation committee based on the number of deposits, a user can easily distribute his/her deposits to different accounts to participate in validation committee, and the proportion of users in the committee is only related to the deposits the user has. In this way, a user who has a large amount of deposits will accumulate more wealth over time. Once this trend continues, which makes a user's wealth to be accumulated enough, just relying on the proof of deposit cannot prevent the user from fully grasping the consensus results in the validation committee. Meanwhile, there is no any efficient mechanism to prevent a user having an account with sufficient deposits from participating in the validation committee. Therefore, we also introduce a proof of work mechanism which requires that only the nodes that have calculated the corresponding proof of work puzzle can participate in the validation committee at each moment. It not only ensures that the members of the validation committee we choose rely on the amount of deposits they have, but also prevents the accumulation of rights and interests. It also prevents the nodes with large deposits from controlling the consensus results.

On the other hand, if we only use proof of work to make judgments here, there is no motivation for ordinary users to join the validation committee and report the verification results correctly. Even when a wrong result has been reported, a user has no punishment other than being kicked out of the committee. Each user in the validation committee is required to provide a deposit as an incentive for honest users who report correct verification results and as a punishment for malicious users who report incorrect verification results. The deposit of malicious users is also used as a reward for the users that report correctly so as to stimulating their honest behaviors.

We assume that for the nodes who want to join the validation committee, the combined hash power of the malicious nodes should be less than $1/4$ of the total hash power at any time. Otherwise, there will be selfish-mining attacks [45]. Suppose a total of w nodes are selected to join the validation committee. If we require that the count of the final validation result from the majority rule exceeds a, it means that the percentage of malicious nodes in the validation committee is enforced to be less than $t = \frac{w-a}{w}$. We assume that X is a random variable that represents the number of times we pick a Byzantine node in the validation committee. In this way, the maximum number of malicious nodes c is equal to $\lfloor w \times t \rfloor$. We can use the cumulative binomial distribution to calculate the probability P when the proportion of malicious node is less than t in the committee of w nodes [46].

$$P[X \leq c] = \sum_{k=0}^{c} \binom{w}{k} p^k (1-p)^{w-k}. \tag{1}$$

TABLE III
BINOMIAL DISTRIBUTION OF THE PROPORTION OF BYZANTINE NODES AMONG ALL COMMITTEE MEMBERS.

t/w	10	20	50	100
0.7	0.9997	1.0000	1.0000	1.0000
0.6	0.9965	1.0000	1.0000	1.0000
0.5	0.9957	0.9992	1.0000	1.0000
0.4	0.9219	0.9784	0.9937	0.9997
0.3	0.7759	0.8034	0.8369	0.8962

The first column of Table III indicates that the proportion of malicious nodes does not exceed t in the committee, and the first row represents the number of nodes in the committee. The data indicate that the probability that the proportion of malicious nodes does not exceed the proportion of t when the number of nodes in validation committee is determined. When the number of malicious nodes in the committee does not exceed t, the proportion of false validation results provided by malicious nodes will not exceed t, because normal nodes will definitely provide correct validation results. In this way, once the validation result collected by the contract exceeds the proportion of t, the result can be regarded as the final correct result. For example, when we take 10 validation committee members, the probability that the number of malicious nodes does not exceed 1/2 of the total number is 0.99. This means that if the smart contract receives a consistent judgment from more than half of the validation committee members, this result can be considered correct when the proportion of malicious nodes does not exceed 1/2. Since normal nodes will definitely provide correct validation results, the incorrect validation results collected by the contract will not exceed the proportion of the number of malicious nodes in committee. It can also be seen from the table that the higher the number of members in the validation committee, the higher the probability that the contract will get the correct judgment result after collecting enough validations. However, even if the number of members of the validation committee is small, the scheme is still secure in our hypothesis.

C. Fairness

As described in Section V-C1 and Section V-C2, during the process of our protocol, whether $\mathcal{C}_1$ or $\mathcal{C}_2$ fails to provide the deposit within the specified time or the transfer of payer $\mathcal{A}$ times out or fails, the contract will return the deposit of $\mathcal{C}_1$ or $\mathcal{C}_2$ and terminate the protocol. Therefore, the participants of honest behavior will not have any loss in this process. We cannot guarantee that the intermediary is completely reliable after being selected, but it can be guaranteed that even if the intermediary exhibits malicious behavior, it will not cause any loss to both parties of the cross-chain transaction.

On the one hand, for the intermediary $\mathcal{C}_1$, the only malicious behavior he/she can perform is to pretend that the transfer failed after obtaining $\mathcal{A}$'s transfer, thereby terminating the contract and obtaining $\mathcal{A}$'s transfer. But in our protocol, the verification result of the transfer transaction between $\mathcal{A}$ and $\mathcal{C}_1$ is secured by the validation committee. Once $\mathcal{C}_1$ provides the deposit and the account address for trading with $\mathcal{A}$, his/her behavior is completely controlled by the contract. If the contract receives the verification result from the validation committee that the transfer is successful, the subsequent operations will be performed automatically. Otherwise, if the transfer result is a failure or exceeds the timer range, the contract will return the deposits of the two intermediaries and terminate the operation.

On the other hand, if $\mathcal{C}_1$ is honest but $\mathcal{C}_2$ is malicious and does not transfer to $\mathcal{B}$, contract will transfer the deposit of $\mathcal{C}_2$ to $\mathcal{B}$. Because we require that no matter whether it is $\mathcal{C}_1$ or $\mathcal{C}_2$, the actual deposit amount in the contract is not less than the actual amount transferred by $\mathcal{A}$ and $\mathcal{B}$. Therefore, after transferring the deposit of $\mathcal{C}_2$ to $\mathcal{B}$, $\mathcal{B}$ can also use this deposit to exchange for the cryptocurrency he/she needs through our scheme. In this case, $\mathcal{A}$ has successfully transferred the cryptocurrency to $\mathcal{C}_1$. Thus, in order to prevent $\mathcal{A}$ from getting extra cryptocurrency in this process, we choose to send the deposit of $\mathcal{C}_1$ to $\mathcal{C}_2$. In this way, $\mathcal{A}$ has completed the normal plan process and has not lost any benefits. On the whole, our cross-cryptocurrency transaction scheme has been successfully carried out for $\mathcal{A}$ and $\mathcal{C}_1$. For $\mathcal{C}_2$ there is an abnormal transaction recorded in the smart contract. But for $\mathcal{B}$ it is necessary to complete the conversion from the deposit to other types of currencies he/she needs. Compared with the normal process, this does bring in extra overhead. But it can ensure that the program proceeds normally. If the scheme is terminated, it will bring no less than the cost of the existing scheme. Because the cryptocurrency that $\mathcal{A}$ has transferred to $\mathcal{C}_1$ is no longer refundable, we cannot guarantee security by only ending the program. One possible solution is to return the deposit of $\mathcal{C}_1$ to $\mathcal{A}$, but this will also bring in additional overhead not less than our existing solution.

D. Stability

Under security assumptions described in Section IV-B, our protocol can well withstand common attacks in blockchain networks.

1) Double-Spend Attack: In our scheme, there is a situation where the payer entrusts two different intermediaries to make the same payment to different users at the same time. Or in the process of entrusting the intermediary, the payer may make the same payment on other occasions. In order to prevent double-spending attack [47], our scheme requires the verifier to comply with the validation method of the cryptocurrency itself. For example, if the payer pays in bitcoin, the verifier must wait for more than 6 blocks after the transaction before sending its judgment to the contract. Our scheme does not make any changes to the cryptocurrency involved to ensure the security of cryptocurrency transactions.

2) Sybil Attack: Any user can send a transaction request, even if the user does not make subsequent transfers after sending the application to the contract. In this case, only a small amount of the Ethereum transaction fees will be lost. During this process, malicious users could send a large amount of small transaction requests and may even refuse to initiate a transaction after requests are sent. However, in our scheme, the intermediary is allowed to make a selection after receiving the user's transaction request. That is to say, the intermediary can confirm that the account has sufficient balance through the user's account address. The intermediary will remove the user request if the transfer amount is too small, thereby ensuring that the user is not a Sybil node [48].

3) DoS Attack: A malicious node may forge a large number of accounts trying to join the validation committee or pretend to be an intermediary in order to disrupt the normal protocol process through DoS attacks. However, our scheme will not be affected by these malicious behaviors since we have two preventive measures. First, each member of the validation committee should pay some ethers as security deposit. Second, each node needs to submit a solution to the PoW puzzle in order to be recognized as a member of the validation committee. Therefore, a malicious node is not able to easily forge its identity to join the validation committee without paying any cost. Likewise, a malicious node is not able to act as an intermediary because the intermediary node needs to have a record of having joined a validation committee.

E. Atomicity

The atomicity of the transaction requires that the cross-chain transaction must be achieved either completely successful or completely failed. And in our scheme there will be no situation where one party pays and the other party does not receive

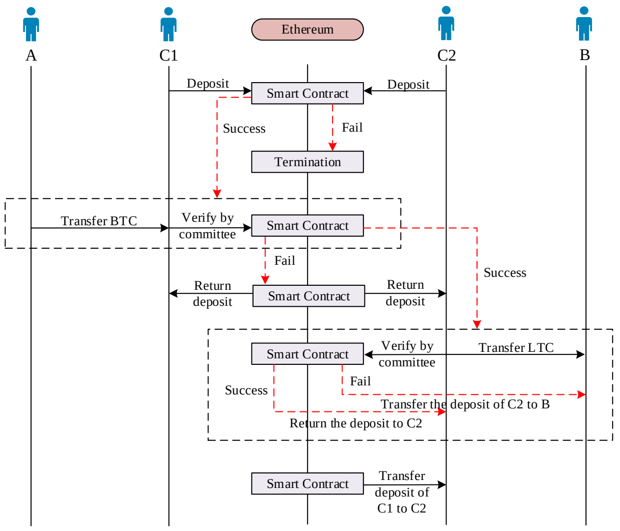


Fig. 3. Protocol state transition diagram.

the transfer. Our scheme achieves this goal through protocol design based on smart contracts and overtime judgment.

In our scheme, if all participants follow the protocol normally, the transaction will be executed properly. If some alliances deviate from the agreement, there will be no losses to the participants in the normal implementation. And the protocol will be terminated or the violation of the operation will be punished. As shown in Fig. 3, our protocol has three execution judgments. The first one is that the intermediary needs to provide a sufficient amount of deposit at the beginning and set a hash time lock on the smart contract. Then, $\mathcal{A}$ can transfer to $\mathcal{C}_1$. Here we rely on the validation committee to judge the transaction results and set a time lock in contract to prevent the transfer from overtime. If the transfer fails, the transaction is terminated. Finally, the intermediary $\mathcal{C}_2$ is required to transfer funds to $\mathcal{B}$, and the validation committee also verifies the transaction results and sets a hash time lock to prevent the transfer from overtime. In case $\mathcal{C}_2$ is malicious and does not transfer to $\mathcal{B}$, contract will transfer the deposit of $\mathcal{C}_2$ to $\mathcal{B}$. Therefore, $\mathcal{B}$ will receive a transfer currency not less than his/her due value. In this case, the party that conforms to the protocol will not suffer losses, and thus the malicious party will not gain any benefit. Thus, there is no incentive for a malicious party to deviate from our protocol.

F. Safety and Liveness

Safety: honest (non-Byzantine) nodes agree on the same value.

Proof: The proposed protocol provides safety if all honest nodes agree on the same result of the transaction validation. To achieve this, the majority rule is leveraged to determine the final validation result. For a transaction tx to be validated, each validation node P_i will give one of the two results: *Validation*(tx, P_i)=$true$ or *Validation*(tx, P_i)=$false$. The conflicting validation results divide the validation nodes into two groups: True(tx)=$\{P_i$, *Validation*(tx, P_i)=$true\}$ and False(tx)=$\{P_j$, *Validation*(tx, P_j)=$false\}$. By using the majority rule, the final validation result will be $true$ if $|$True(tx)$|>|$False(tx)$|$; Otherwise, i.e., if $|$False(tx)$|>|$True(tx)$|$, it will be $false$. According to our security assumption that the number of Byzantine nodes is less than 1/4 in the committee, nodes who give the final validation result are honest, and others are Byzantine. We choose to prove this feature by contradiction. We are to show that it's not possible that $true$ and $false$ are both recognized by the committee according to the majority rule. For the recognition of the final result of $true$, it has to be the case that "$true$" has been recognized by more than half of the nodes in the committee. Meanwhile, for the recognition of the final result of $false$, it has to be the case that "$false$" has also been recognized by more than half of the nodes in the committee. However, this implies that $|| \ True(tx) \ | + | \ False(tx) \ ||$ is larger than the total number of the members in the committee, which is impossible. Therefore, this creates a contradiction, as neither of the two results cannot be recognized by more than half of all of the nodes in the committee at the same time. In other word, honest nodes do not agree to these two different results, thereby proving that our protocol achieves safety.

Liveness: transaction will eventually abort or commit.

Proof: In our scheme, we adopt the time assumption of weak/partial synchronization, which means messages are guaranteed to be delivered after a certain time bound. The design of our scheme is also based on this assumption. As a result, our scheme will set a timer in the smart contract to prevent malicious nodes from refusing to provide validation result. Under this weak/partial synchronized time assumption, in order of execution, there are five kinds of requests in the contract: $\mathcal{R}_1$(C_1&C_2 deposit), $\mathcal{R}_2$(payer transfer), $\mathcal{R}_3$(payer confirm), $\mathcal{R}_4$(payee transfer), and $\mathcal{R}_5$(payee confirm). Accordingly, there are six valid states: $\mathcal{S}_0$(start), $\mathcal{S}_1$(intermediaries deposit), $\mathcal{S}_2$(payer transferred), $\mathcal{S}_3$(payer confirmed), $\mathcal{S}_4$(payee transferred), S_5(payee confirmed). At each state $\mathcal{S}_i$, the contract waits for the corresponding request R_{i+1}. If successful, the contract then transfers to the next state $\mathcal{S}_{i+1}$; Otherwise, if failed or timeout, the contract then terminates and declares the transaction's failure. If the contract successfully traverses the valid state transition sequence, i.e. "$\mathcal{S}_0 \rightarrow \mathcal{R}_1 \rightarrow \mathcal{S}_1 \rightarrow \mathcal{R}_2 \rightarrow \mathcal{S}_2 \rightarrow \mathcal{R}_3 \rightarrow \mathcal{S}_3 \rightarrow \mathcal{R}_4 \rightarrow \mathcal{S}_4 \rightarrow \mathcal{R}_5 \rightarrow \mathcal{S}_5$", the transaction is then successfully completed. In conclusion, the contract always makes progress and will never block indefinitely at any state. Thus, our protocol achieves liveness.

VII. Performance Analysis

To evaluate the deployment cost of our scheme, we deploy our contract on the Ethereum's official test network Ropsten [49]. Gas is used to measure the cost of each step in the smart contract code. Gas and financial costs of each of functions described in Section V-B are outlined in Table IV. We have approximated the cost in USD ($) using the conversion rate of 1 ether = $130 and the gas price of 0.000000003 ethers which are the real cost in April 2020. The code for the intermediaries to calculate the PoW puzzle is not included in the contract because users can execute this part on their local machine and send the results to the contract.

TABLE IV
THE GAS VALUE OF EACH OPERATION.

Transaction	Gas	USD
Deploy	3690283	1.43
IC.Register	181591	0.07
IC.Update	80995	0.03
CC.Verify_PoW	191737	0.07
TC.Prepare	398525	0.15
TC.Deposit	36452	0.01
TC.Validation	163780	0.06

For users who need to make cross-currency transactions, only the function of *TC.Prepare* needs to be invoked by the user. Even for the multi-party situation, each user only needs to call this function once, which costs \$0.15. For intermediate users, there are *IC.Register* and *IC.Update* functions to provide their own information, which cost only \$0.07 and \$0.03, respectively. In a transaction, the operation of issuing deposit for each intermediary costs \$0.01 on average. The intermediary who wants to join the validation group needs to perform the validation operation of *CC.Verify_PoW* costing \$0.07. The cost of validation is borne by every member of validation group. Depending on the invocation overhead of our contract, users can choose whether to use our service within acceptable limits.

A. Timing Analysis

We also measure the runtime of our contract. All measurements were performed on a desktop running Windows 10 equipped with 6 cores, 3.0 GHz Intel Core i5 and 8 GB DDR4 RAM. In this section, we will present the runtime results for our scheme, which shows that our contract can achieve the expected functionality within an acceptable time overhead.

The implementation of our scheme is divided into two parts: off-chain part and on-chain part. The calculation of PoW puzzle performed by users to join the verification group is performed off-chain on the user's local machine. The smart contract only needs to receive and verify the solution of the PoW puzzle through *IC.Verify_PoW* function and adjust the difficulty value of PoW according to the solution time. In this way, the time interval for solving the PoW problem can be kept at around ten minutes. For the on-chain part, we only consider the operating time of the smart contract in the scheme. The scheme involves the confirmation time of other types of cryptocurrencies on its blockchain, which is not our consideration.

Due to the limitation of gas and block size, we set the number of payers and payees involved in the transaction to be less than ten in a round of protocol process. When we change the number of users participating in the transaction, the execution time of the preparation phase of the contract will be mainly affected. We record the largest execution time of the users who independently call the contract through *TC.Prepare* function and show the results in Fig. 4. As we can see, the runtime of our contract increases as the number of payees increases. This is because for a single payer, the contract needs to record and process the information of each payee of the user. Thus, the larger the number of all payees, the more information will be sent to the contract, which leads to a longer overall runtime of the contract. It can be also seen that since each of the payers is a different node and their operations are parallel. Increasing the number of payers does not significantly increase the contract runtime. Thus, the preparation time of the contract is only affected by the number of transaction users of each payer. After collecting ten users' requests or exceeding this waiting time, the contract will proceed to the next step.

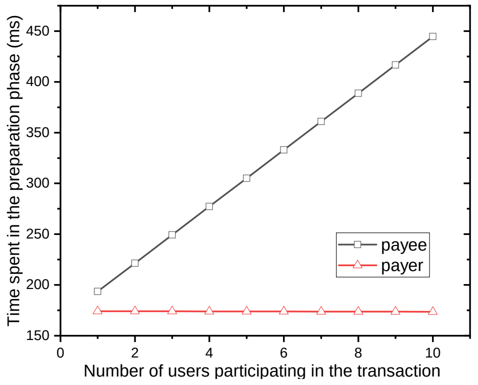


Fig. 4. Time spent on the preparatory phase when the number of payees increases.

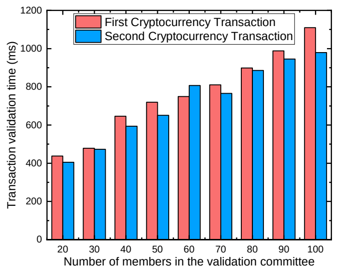


Fig. 5. Time spent on the validation phase when the size of validation committee increases.

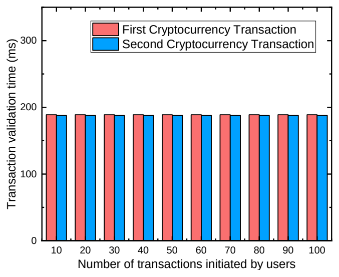


Fig. 6. Time spent on the validation phase when the number of transactions between a pair of users changes.

We also change the number of users in the validation committee and the number of transactions between a pair of users, so as to analyze the performance of the contract in the validation process. When we change the number of participants in the validation committee, the time which the contract takes to perform this step through *TC.Validation* function is shown in Fig. 5. The first validation is for the transaction between payers and intermediary $\mathcal{C}_1$, while the second validation is for the transaction between intermediary $\mathcal{C}_2$ and payees. As we can see, when the number of validation committee members increases, the time required for the validation process also increases significantly. Nevertheless, even with 100 members in the validation committee, each validation process takes only about 1 second. This timing is negligible compared to the time it takes for transactions of public blockchain cryptocurrency like bitcoin to be confirmed. We can also see from Fig. 6 that as the number of transactions between a payer and a single payee increases, it will not increase the time cost of transaction verification through the contract. This is because all transactions between the same pair of users will be merged into one transaction through the contract, thereby not increasing the time cost.

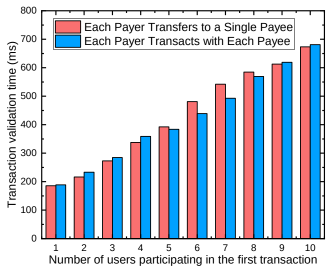


Fig. 7. Time spent on the validation process for the first cryptocurrency transaction.

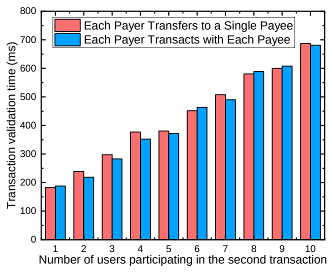


Fig. 8. Time spent on the validation process for the second cryptocurrency transaction.

When we increase the number of users involved in the transaction, each additional transaction for each additional user adds to the runtime cost of the contract. For the convenience of the experiment, we add a pair of payer and payee at the same time. Fig. 7 and Fig. 8 show the validation time required in two transactions respectively. The two colored columns in both figures represent two different cases. The first is the case where every payer transacts with a single payee, in which the number of transactions equals the number of participants. While the second case is when each payer transacts with each payee, in which the number of transactions is twice as large as the number of participants. However, in the range of allowable error, we can conclude from the figure that the time cost is the same in both cases. This means that the validation time of our scheme is only affected by the number of users involved in the transaction, regardless of the number of transactions per user. This is because in our contract, the transactions involved in each participating user are merged, which is also the premise of our solution to maintain high throughput in the presence of a large number of transactions.

We also measure the verification time required to call the main functions of the contract at different times. As shown in Fig. 9, although the validation time on the test chain can only be used as a reference, it can be seen that the contract execution time is negligible compared to the transaction confirmation time. Therefore, the execution time on the machine required by our solution is acceptable. In future work, we will also deploy our contract on the Ethereum main chain for testing under the actual network.

VIII. Conclusion

In this paper, we proposed a decentralized cross-cryptocurrency exchange scheme for transactions among multiple users based on smart contracts. In our scheme, we use ether as a transit to link transactions between different kinds of cryptocurrencies. We also implemented the contract and evaluated its execution overhead on our local machine. The results show that the time cost of our scheme is only affected by the total number of users involved in the transaction while the number of user's transactions has no significant impact on the runtime of our contract. In the future, we will further improve our scheme from the experimental part in two aspects.

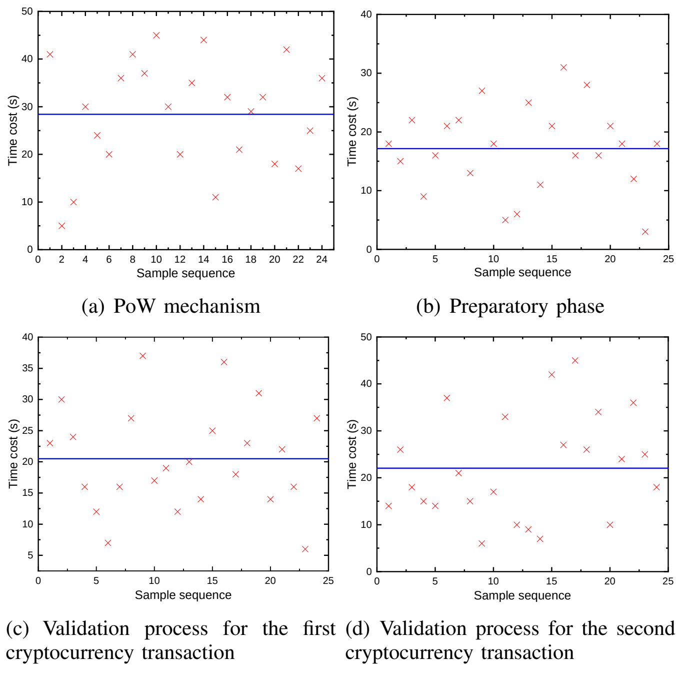


Fig. 9. Time cost of transaction validation in the main steps of protocol

Firstly, we will try to complete the implementation of the scheme with more participants. Secondly, we will deploy our scheme on the Ethereum Mainnet to test the cost of our scheme in the actual network.

Acknowledgment

The authors sincerely thank the editor, Dr. Debdeep Mukhopadhyay, and the anonymous referees for their valuable suggestions that have led to the present improved version of the original manuscript. This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61972371 and Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, accessed on Apr., 2021. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] C. Lee, "Litecoin," https://litecoin.org/, 2011, accessed on Apr., 2021.
[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger EIP-150 REVISION (759dccd-2017-08-07)," 2017, Ethereum Project Yellow Paper, Accessed on Apr., 2021. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf
[4] "P2PB2B," https://p2pb2b.io, accessed on Apr., 2021.
[5] "MXC," https://mxc-exchange.zendesk.com/hc/zh-cn, accessed on Apr., 2021.
[6] "BKEX," https://www.bkex.co, accessed on Apr., 2021.
[7] "Bilaxy," https://bilaxy.com, accessed on Apr., 2021.
[8] "LBank," https://www.lbank.info, accessed on Apr., 2021.
[9] C. Y. Kim and K. Lee, "Risk management to cryptocurrency exchange and investors guidelines to prevent potential threats," in *Proceedings of the 2018 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2018, pp. 1–6.

[10] R. Khalil, A. Gervais, and G. Felley, "TEX-A securely scalable trustless exchange," 2019, cryptology ePrint Archive. [Online]. Available: https://eprint.iacr.org/2019/265

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

[12] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016, pp. 254–269.

[13] V. Buterin, "A next-generation smart contract and decentralized application platform," 2014, white paper, Accessed on Apr., 2021. [Online]. Available: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

[14] "Metronome," https://www.metronome.io, accessed on Apr., 2021.

[15] Y. V. L. Luu, "Kybernetwork: A trustless decentralized exchange and payment service," 2018, white paper, Accessed on Apr., 2021. [Online]. Available: https://whitepaper.io/document/43/kyber-network-whitepaper

[16] J. Chow, "BTC relay," https://github.com/ethereum/btcrelay, accessed on Apr., 2021.

[17] F. Vogelsteller and V. Buterin, "EIP 20: ERC-20 token standard," 2015, accessed on Apr., 2021. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20

[18] W. Warren and A. Bandeali, "0x: An open protocol for decentralized exchange on the ethereum blockchain," 2017, white paper, Accessed on Apr., 2021. [Online]. Available: https://github.com/0xProject/whitepaper

[19] "EtherDelta," https://etherdelta.com, accessed on Apr., 2021.

[20] "IDEX," https://idex.market, accessed on Apr., 2021.

[21] "TBTC: A decentralized redeemable btc-backed erc-20 token." 2019, accessed on Apr., 2021. [Online]. Available: https://docs.keep.network/tbtc/index.pdf

[22] T. Zhang and L. Wang, "Republic protocol," 2017, accessed on Apr., 2021. [Online]. Available: https://republicprotocol.github.io/whitepaper/republic-whitepaper.pdf

[23] J. Teutsch, M. Straka, and D. Boneh, "Retrofitting a two-way peg between blockchains," 2019, arXiv preprint. [Online]. Available: https://arxiv.org/pdf/1908.03999

[24] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," 2016, white paper, Accessed on Apr., 2021. [Online]. Available: https://cosmos.network/whitepaper

[25] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," 2016, white paper, Accessed on Apr., 2021. [Online]. Available: https://github.com/w3f/polkadot-white-paper/raw/master/PolkaDotPaper.pdf

[26] J. Kwon, "Tendermint: Consensus without mining, Draft v.0.6," 2014, white paper, Accessed on Apr., 2021. [Online]. Available: https://tendermint.com/docs/tendermint.pdf

[27] A. Zamyatin, D. Harz, J. Lind *et al.*, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 193–210.

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

[29] S. Matetic, M. Ahmed, K. Kostiainen *et al.*, "{ROTE}: Rollback protection for trusted execution," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*, 2017, pp. 1289–1306.

[30] N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza, "AsyncShock: Exploiting synchronisation bugs in intel SGX enclaves," in *Proceedings of the 2016 European Symposium on Research in Computer Security (ESORICS)*. Springer, 2016, pp. 440–457.

[31] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 2018, pp. 245–254.

[32] M. Herlihy, B. Liskov, and L. Shrira, "Cross-chain deals and adversarial commerce," *Proceedings of the VLDB Endowment*, vol. 13, no. 2, pp. 100–113, 2019.

[33] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind *et al.*, "Anonymous multi-hop locks for Blockchain scalability and interoperability," in *Proceedings of the 2019 Annual Network & Distributed System Security Symposium (NDSS)*, 2019.

[34] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A^2L: Anonymous atomic locks for scalability and interoperability in payment channel hubs." 2019, cryptology ePrint Archive. [Online]. Available: https://eprint.iacr.org/2019/589.pdf

[35] P. Moreno-Sanchez, A. Blue, D. V. Le *et al.*, "DLSAG: Non-interactive refund transactions for interoperable payment channels in Monero," in *Proceedings of the 2020 International Conference on Financial Cryptography and Data Security (FC)*, 2019.

[36] 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 (CCS)*, 2019, pp. 801–815.

[37] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2012, self-published paper, Accessed on Apr., 2021. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[38] "EOS.IO technical white paper v2," 2018, white paper, Accessed on Apr., 2021. [Online]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[39] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Proceedings of the 2015 International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2015, pp. 507–527.

[40] M. Castro, B. Liskov *et al.*, "Practical Byzantine fault tolerance," in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI)*, vol. 99, no. 1999, 1999, pp. 173–186.

[41] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[42] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Annual Technical Conference (ATC)*, 2014, pp. 305–319.

[43] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, 2018, pp. 245–254.

[44] A. Zamyatin, M. Al-Bassam, D. Zindros *et al.*, "SoK: Communication across distributed ledgers," 2019, iACR Cryptology ePrint Archive. [Online]. Available: https://eprint.iacr.org/2019/1128.pdf

[45] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proceedings of the 2014 International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2014, pp. 436–454.

[46] E. K. Kogias, P. Jovanovic, N. Gailly *et al.*, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proceedings of the 25th USENIX security symposium (USENIX Security)*, 2016, pp. 279–296.

[47] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and Communications Security (CCS)*. ACM, 2012, pp. 906–917.

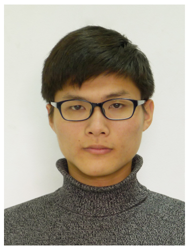
[48] J. R. Douceur, "The sybil attack," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*. Springer, 2002, pp. 251–260.

[49] "The ethereum block explorer: ROPSTEN (Revival) TESTNET," https://ropsten.etherscan.io, accessed on Apr., 2021.

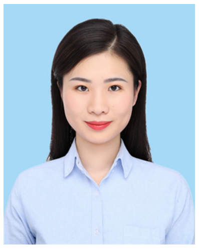
Hangyu Tian received his bachelor's degree from the department of Information Security, University of Science and Technology of China (USTC) in July, 2018. He is currently a graduate student in Communication and Information System from the Department of Electronic Engineering and Information Science(EEIS), USTC. His research interests include Network security and Cryptography.

Kaiping Xue (M'09-SM'15) received his bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Security and the Department of EEIS, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. He serves on the Editorial Board of several journals, including the IEEE Transactions on Dependable and Secure Computing (TDSC), the IEEE Transactions on Wireless Communications (TWC), and the IEEE Transactions on Network and Service Management (TNSM). He has also served as a guest editor of IEEE Journal on Selected Areas in Communications (JSAC), and a lead guest editor of IEEE Communications Magazine and IEEE Network. He is an IET Fellow and an IEEE Senior Member. He is the corresponding author of this paper.

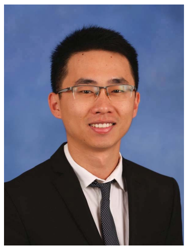
Xinyi Luo received her B.S. degree in Information Security from School of the Gifted Young, University of Science and Technology of China (USTC) in July, 2020. She is currently a graduated student in School of Cyber Security, USTC. Her research interests include Network security and Cryptography.



Shaohua Li received the B.E. degree from the Department of Information Security, University of Science and Technology of China (USTC) in 2016, and received his M.S. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2019. He is currently a Ph.D. student in Department of Computer Science in ETH Zurich, Switzerland. His research interests include network security protocol design and analysis.



Jie Xu received her B.S. degree from the department of Information Security, University of Science and Technology of China (USTC) in July, 2017, and received her M.S. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2020. She is currently a Ph.D. student in the Department of Computer Science, City University of Hong Kong. Her research interests include Network security and Cryptography.



Jianqing Liu (M'18) received his B.Eng. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2013 and received his Ph.D. degree with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA, in 2018. Currently, He is an assistant professor in the Department of Electrical and Computer Engineering at the University of Alabama in Huntsville. His research interests include wireless networking and network security in cyber-physical systems.

Jun Zhao (S'10-CM'15) received the bachelor's degree from Shanghai Jiao Tong University, Shanghai, China, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore. Before joining NTU first as a Postdoc with Xiaokui Xiao and then as a faculty member, he was a Postdoc with Arizona State University as an Arizona Computing PostDoc Best Practices Fellow. His research interests include blockchain, security, and privacy with applications to the Internet of Things and deep learning.

David S.L. Wei (SM'07) received his Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. From May. 1993 to Aug. 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan. He is currently a Professor of Computer and Information Science Department at Fordham University. His research interests include cloud and edge computing, machine learning, IoT, and information systems security. He was a lead guest editor or a guest editor for several special issues in the IEEE Journal on Selected Areas in Communications, the IEEE Transactions on Cloud Computing, and the IEEE Transactions on Big Data. He also served as an Associate Editor of the IEEE Transactions on Cloud Computing, 2014-2018, and IEEE J-SAC for the Series on Network Softwarization & Enablers (2018-2020).