Cross-chain Exchange by Transaction Dependence with Conditional Transaction

Hong Su

Abstract—Cross-chain exchange swaps assets among different blockchains, which facilitates cooperation among blockchains. A cross-chain exchange contains several transactions from different blockchains. It requires to synchronize asset transfers in those associated blockchains to avoid partial transfers. Current cross-chain methods are divided into two main types. One locks the assets first and transfers the frozen asset to receivers later. The second one is the two-phase or three-phase transaction commit protocol. Those methods require at least two steps, which makes those blockchains coherent to synchronize at different time(steps). Serialization is required as the second step has to wait for the completion of the first step, and even some steps in the same stage are required to be serialized. Meanwhile, their implements are either by smart contracts or special blockchain structures/roles. Smart contract based methods require to pre-deploy a smart contract and cannot change dynamically. Special structure or role-based methods force associated blockchains to have those special requirements. In this paper, we propose a new cross-chain exchange model based on the dependence of associated transactions, which is expressed inside a transaction and can change when sending transactions. It saves the step to lock the asset or perform a pre-commit and has no special requirements for blockchains or pre-deployment of smart contracts. The simulation results show the proposed model exchanges asset effectively among different blockchains.

Index Terms—Cross-chain, cross-chain exchange, transaction dependence, conditional transaction.

I. INTRODUCTION

BLOCKCHAIN guarantees the security of digital assets. It does not rely on a trusted third party. This makes it different from the central authority currency system [1]. Blockchain technology has been successfully applied to the digital currency area, such as Bitcoin [1], Ethereum coins [2], Namecoin [3]. Except for applications in the digital currency area, blockchain is also applied to other fields [4] [5]. The application areas include food supply improvement [6] [7] [8], contributions to smart city [9] [10] [11], securing distributed database log [12], and contacts between people [13].

Those applications are deployed on different kinds of blockchains. It is important to interact with different blockchains, which form a "big" blockchain network to connect valuable things, like the Internet to connect small networks[14] [15]. Valuable assets are exchanged directly among those blockchains, which saves extra steps for users to identify a third party exchange agency. Thus, cross-chain interaction has been introduced [16] [15]. One of the cross-chain interactions is the cross-chain exchange, which swaps different assets among several blockchains.

H. Su is with the Department of Computer Science, Sichuan University, Chengdu, China.

Fig. 1. Cross-chain exchange between two blockchains.

In an exchange, cross-chain transactions have dependencies on each other. The transaction in one blockchain requires associated transactions in other blockchains. For example, user 1 has some digital coins in blockchain A. It wants some asset on blockchain B. One possible solution is to find one user (user 2) on blockchain B who wants to exchange its asset with digital coins in blockchain A. User 1 transfers some amount of digital coins in blockchain A (state change of blockchain A); user 2 transfers the corresponding amount of digital asset to user 1 (state change of blockchain B). Figure 1 shows the Cross-chain exchange between two blockchains.

A cross-chain exchange contains several transactions from different blockchains, which requires to synchronize asset transactions in all associated blockchains and to avoid partial transactions. For example, there are two transactions in an exchange shown in Figure 1. One transfers asset 1 from user 1 to user 2, and another transfers asset 2 from user 2 to user 1. With those two transactions, each one uses what it has to obtain what it desires on a different blockchain. If there is only one transfer in a blockchain, the receiver in another blockchain cannot obtain its asset.

The methods to ensure the atomicity are divided into two main types. One locks assets into a special account or a smart contract in the first stage, and in the second stage, assets are allocated to receivers. Another adopts the special execution order, including two-phase or three-phase commit. Specifically, some methods adopt special blockchain roles [16] or special structures [17] to lock the asset or ensure the execution order. As they have those requirements for the blockchain, it is difficult to apply to blockchains without the required roles or structures. Other methods adopt the smart

contract to hold the locked asset or keep the execution order. It has a requirement to pre-deploy smart contracts on associated blockchains and cannot be changed when an exchange is undergoing.

Those methods are executed in two steps. Asset locking methods lock assets first, and later transfer locked assets to receivers. The two-phase or three-phase commit methods have explicit pre-commit and commit steps. The two-step methods bring disadvantages. (1) It requires different blockchains to cooperate those steps which happen at different time. It brings the time coherence among corresponding blockchains. (2) It is required for the second step to wait for the completion of the first steps. The waiting time lowers the cross-chain transaction process efficiency.

In this paper, we propose a new cross-chain exchange model, which is based on the dependence of associated transactions in an exchange. To express the transaction dependence, we introduce the dependence keyword. A new field (condition field) is added to the blockchain transaction data structure (we call this kind of transaction the conditional transaction) to contain the transaction dependence. The asset is locked when the transaction is sealed into the blockchain, and it is automatically transferred to receivers when the exchange completes.

The major contributions of this paper are as follows. (1) We propose the cross-chain exchange model, which only needs one explicit step (users to send conditional transactions) and does not need a specific execution order of associated transactions. Meanwhile, it is directly based on the transaction itself instead of other aspects of the transaction, which makes it possible to use the transaction data fully. (2) We propose the cross-chain conditional transaction to express the dependence of cross-chain transactions. The dependence is inside the transaction and has no requirement to pre-deploy a smart contract. (3) We propose to use the condition keyword to specify the relationship between transactions. It can describe the different combinations of dependent transactions.

The rest of the paper is organized as follows. Section 2 describes the cross-chain exchange model. Section 3 describes the implementation of cross-chain exchange, the conditional transaction. Section 4 shows the verification results. Section 5 describes the related work, and section 6 concludes the paper.

II. CROSS-CHAIN EXCHANGE

This section describes the cross-chain exchange model. The motivation is introduced first. Then, the transaction dependence in an exchange is discussed. Based on the transaction dependence, we describe the exchange model. At last, the security of this model is analyzed.

A. Motivation

Current methods consist of two explicit steps [18] [19]. However, it is unnecessary to have two steps in a cross-chain exchange. If all associated transactions appear, the exchange can be automatically performed. And the consistency of the transactions is ensured by blockchains with their consensus algorithm.

A one-step exchange brings benefits. It is more feasible as there is no need to cooperate two separate exchange steps across different blockchains, and there are no constraints on a specific order to lock or unlock the assets. Comparing to current cross-chain methods, the sequence of the steps is necessary to ensure the correctness of the exchange, such as in hash-locking[23] or notary method[25]. The one-step exchange saves the time to wait for the completion of the previous steps.

The one-step exchange facilitates the usage of the cross-chain exchange. Users only need to send a special transaction (referring to section III), which has additional parameters to specify dependent transactions.

There is no need to deploy special smart contracts in different blockchains. Cross-chain smart contract deployment requires all blockchains to support the same smart contract language, or it requires to program the same logic in different smart contract languages for different blockchains.

B. Cross-chain Exchange and Cross-chain Transaction Dependence

Cross-chain exchange is a process in which one user uses its digital asset in a blockchain to exchange other assets in different blockchains. The aim of the cross-chain exchange is to ensure that one gets the asset in a blockchain after it pays in another blockchain. Each transfer is a transaction from a blockchain. Thus, a cross-chain exchange (CCE) is composed of transactions from different blockchains, referring to formula (1).

$$\text{CCE} = \{tx_{1k}, \dots, tx_{ij}, \dots, tx_{nm}\} \quad (1)$$

, where tx_{ij} is transaction i from blockchain j .

Transactions depend on other transactions in the same exchange, as a participant transfers its asset in a transaction to another one with the aim of obtaining the asset from other transactions. Suppose there are two transactions, tx_{mn} , tx_{kl} in an exchange. We see that if any of them is missing, the exchange does not complete. Thus, transaction tx_{mn} in blockchain n depends on transaction tx_{kl} in blockchain l . Furthermore, they depend mutually, i.e. tx_{kl} also depends on tx_{mn} .

We introduce the dependence function dep to describe the dependence between cross-chain transactions. Then the relationship of mutual dependence is shown in (2).

$$\begin{aligned} &\text{When } tx_{mn}, tx_{kl} \in \text{CCE} \\ &\text{if } tx_{mn} \text{ dep } tx_{kl} \text{ then } tx_{kl} \text{ dep } tx_{mn} \end{aligned} \quad (2)$$

We only discuss the case when $l \neq n$; otherwise, it is an exchange in the same blockchain.

1) *Transaction Dependence Keyword*: The transaction dependence keyword helps to express the transaction dependence in an exchange. Associated blockchains are only required to support those keywords; instead of the requirement to pre-deploy a smart contract.

Keywords can be classified into decision-making keywords and cross-chain keywords. The decision-making keywords

TABLE I
CONDITION KEYWORD

Decision-making keyword	Cross-chain keyword
IF ELSE AND OR	doesTxExpect (tx, chainID, ...)
...	...

have a similar semantic meaning as keywords in C programming language. The cross-chain keywords are used to judge or set the requirement for cross-chain transactions. Judgment functions (including `doesTxExpect`) judge whether a cross-chain transaction is the expected one or not. Table I lists the condition keywords used in this paper.

With those keywords, it is flexible to express the cross-chain condition. If tx_1 and tx_2 form an exchange, the dependence fields of tx_1 is 'IF `doesTxExpect` (tx_2 , ...)'. It is also used to describe the combination of transaction dependence. For example, we can use the keyword 'AND' to express the dependence on more than one transaction. Suppose tx_1 , tx_2 , and tx_3 form an exchange, the dependence fields of tx_1 is 'IF `doesTxExpect` (tx_2 , ...) AND `doesTxExpect` (tx_3 , ...)'.
 2) *Different Types of Cross-chain Dependence among Blockchains*: A cross-chain exchange is not limited to only two blockchains, which can be among several blockchains. We describe two typical blockchain dependence types.

The first one is one-to-one blockchain dependence. One transaction on a blockchain depends on another one on a different blockchain. The dependent transaction further depends on another one. This dependence iterates on to form a dependence cycle among several blockchains. Figure 2 illustrates an example.

Fig. 2. One to one dependence. The notation of 'tx' means transaction.

The second one is that one transaction on a blockchain depends on transactions of several other blockchains. Figure 3 shows an example of this case. Blockchain 1 can be seen as the centralized blockchain, which other blockchains exchange directly. Other blockchains (for example, blockchains 2 and 4) can exchange indirectly with blockchain 1 as the intermediate.

Fig. 3. One blockchain transaction exchanges with cross-chain transactions on different blockchains. The notation of 'tx' means transaction and the notation of 'BC' means blockchain.

We call this server-client type. The transaction which interacts with several transactions is called the server transaction, and other transactions are called client transactions. The dependence of the server transaction is described in algorithm 1.

Algorithm 1 A transaction requires transactions from several blockchains

```

1: if doesTxExpect(..., chainID1) then
2:   if doesTxExpect(..., chainID2) then
3:     if doesTxExpect(..., chainID...) then
4:       // All required transactions appear.
5:       return true
6:     end if
7:   end if
8: end if
9: // At least one required transaction does not appear.
10: return false

```

C. One-Step Model

The proposed model wait for all dependent transactions and completes when those transactions match. We regard each transaction as the promise to give the asset under the condition that its required transactions appear, also called a promise-condition (or P-C) pair. A promise-condition pair can be the condition of another promise-condition pair. If all promise-condition pairs match, the exchange completes. It only needs one-step to wait for the dependent transactions from other blockchains. There are no extra steps to synchronize or have a second phase commit. It is called the one-step cross-chain exchange model or one-step model, referring to Figure 4.

In this model, the asset is automatically locked when its P-C pair is sent out. When the exchange completes, it automatically transfers to the receiver. It uses the delayed asset transfer method.

1) *Waiting Procedure*: The proposed model needs to wait for all associated transactions. Blockchains synchronize cross-chain transactions among them. In this paper, we skip the details of cross-chain propagation and assume that cross-chain transactions can be propagated to other blockchains.

Figure 5 shows the waiting process, in which exchanges wait for and check the associated cross-chain transactions (such as tx_{kl} , tx_{ml} , tx_{ij}).

Fig. 4. One-step model. It only requires each participant to send a conditional transaction, instead of two steps to lock asset and then to transfer the asset.

Fig. 5. Exchange depends on the associated transactions in each blockchain. The notations of tx_{kl} , tx_{ml} , and so on stand for transactions used for cross-chain exchange.

The waiting process is described in the algorithm 2, which checks dependent transactions from line 2 to line 6. This algorithm begins when new blocks are sealed into the current blockchain. It iterates transactions in those blocks and checks whether it is required by the target transaction (ctx) in line 2. If it matches, it continues to check whether all required transactions of ctx match in line 3, and returns corresponding result in lines of 4 and 8.

To further describe the exchange process, we introduce a variety, CCE_Status , which describes the status in the process to wait for associated transactions. It returns 'true' when all related transactions appear ('appear' means that a transaction has been put to its blockchain and propagated to the checked blockchain), shown in formula (3).

$$CCE_Status = \begin{cases} false, & \text{if } \exists tx_{ij} \in CCE, tx_{ij} \notin BCT \\ true, & \text{otherwise} \end{cases}$$

, where BCT is the collection of transactions which are sealed in blockchain, referring to the next formula.

(3)

Algorithm 2 Judgement of the completion of the Cross-chain Exchange

Require:

newBlockList, new blocks sealed in blockchain;
 ctx , the target cross-chain transaction in current blockchain to judge its completion.

Ensure:

The result (*result*) indicates whether all required transactions matched. Return true when matched; otherwise return false;

```

1: for transaction  $ntx$  in newBlockList do
2:   if  $ntx$  is required by  $ctx$  then
3:     if all required transactions match then
4:       return true
5:     end if
6:   end if
7: end for
8: return false
  
```

$$BCT = \{tx_1, \dots, tx_i, \dots, tx_n\}$$

, where tx_i is the transaction sealed in the blockchain. (4)

In formula (3), the notation of tx_{ij} is the cross-chain transaction i from blockchain j . This is described in formula (5).

$$tx_{ij} = (tx_i, cid_j)$$

, where the pair of (tx_i, cid_j) stands for the transaction tx_i in the blockchain with blockchain id cid_j .

(5)

From formula (3), we see an exchange is impacted by associated blockchains instead of its own blockchain. The reason is that an exchange contains several associated transactions that are from different blockchains. Each transaction needs to be confirmed in its own blockchain and then propagated to other blockchains. All the blockchains have an impact on the exchange.

We take the impact of the mining period to the time of exchange as an example. If there is a big mining period in one blockchain, the exchange time on each blockchain is a big one.

First, the transaction is put to its blockchain. The average time of this process ($t_{toChain}$) is shown in formula (6).

$$t_{toChain} = t_{wait} + t_{mining} + t_{confirm} \quad (6)$$

Formula (6) shows that the time of $t_{toChain}$ is decided by three parameters. (1) The average waiting time (t_{wait}). It is determined by the mining period and the load of its blockchain. (2) The mining period (t_{mining}), which is the average time to seal a block in that blockchain. (3) The average confirmation time ($t_{confirm}$), which is the waiting time to confirm that the transaction does not go to a branch chain. It is also proportional to the mining period. We see that $t_{toChain}$ is specific to a blockchain, which is determined by its mining period and its transaction load.

When a conditional transaction propagates to another blockchain, there is a propagation delay time ($t_{propagation}$), which consists of two components, $t_{original_confirm}$ and $t_{network}$. Among those, $t_{original_confirm}$ is the confirmation time in its original blockchain (the transaction sending blockchain), which is further discussed in section II-D3. We also use $t_{confirm}$ to replace $t_{original_confirm}$ if not conflicting. And $t_{network}$ is the network time for the cross-chain transaction to propagate to another blockchain.

$$t_{propagation} = t_{original_confirm} + t_{network} \quad (7)$$

The total time is shown in formula (8).

$$t_{total} = t_{toChain} + t_{propagation} \quad (8)$$

The exchange complete time ($t_{exchange}$) is the time when the last transaction appears. It is the max time of transactions to appear.

$$t_{exchange} = \max(t_{total_i})$$

, where t_{total_i} stands for t_{total} on the blockchain i .

(9)

Although the exchange time is the maximum one, all blockchains have an impact on this process. Thus, a cross-chain exchange is affected by all associated blockchains.

2) *Exchange Completion*: When the associated transactions appear (referring to (10)), the exchange completes and assets are transferred to their receivers in corresponding blockchains.

$$\forall t_{ij} \in CCE, t_{ij} \in BCT \quad (10)$$

However, blockchains may get the transactions at different time, as the propagation time ($t_{propagation}$) is different for cross-chain transactions. Take Figure 6 as an example. Suppose the first blockchain completes its exchange at t_0 when it gets all cross-chain transactions. Blockchain 1 completes at t_1 which has a d_1 deviation from t_0 , and blockchain n completes at t_n which has a d_n deviation. Those time sequences are in parallel instead of a serialized one, which is an advantage of our method when compared to other methods like hash-locking which is a serialized one [18].

Fig. 6. Asset transfer to receiver in each blockchain when exchange completes.

Exchanges on different blockchains complete at different time, while the exchange in each blockchain obtains the same result, as blockchains keep the immutability of transactions. This can be seen as follows.

Suppose, CCE_k is the exchange on blockchain k . If CCE_k completes at time t_k , all associated transactions are in this blockchain, referring to formula (11).

$$\forall tx_{ij} \in CCE_k, tx_{ij} \in BCT_k \quad (11)$$

Later, in time t_{k+1} when associated transactions arrive blockchain $k+1$, the exchange on blockchain $k+1$ (CCE_{k+1}) completes. All transactions in CCE_{k+1} are also in CCE_k , as they are the same exchange to have same dependent transactions. This is shown in formula (12). We have waited for enough confirmation time ($t_{confirm}$), and then it is a small enough possibility that some transactions have been put to a branch chain. Thus, blockchain $k+1$ obtains the same result. This is the same case for other blockchains in this exchange.

$$\forall tx_{mn} \in CCE_k, tx_{mn} \in CCE_{k+1} \quad (12)$$

D. Security Analysis and Corresponding Solution

In this subsection, we analyze the security when an exchange is among different blockchains.

1) *Assumptions*: The first assumption is about the immutability degree of a blockchain. We assume that a block with more successive blocks is less possible to be re-branched. With this assumption, users can wait for enough blocks to ensure that the possibility of the blockchain re-branch is relatively low to a certain value.

The second assumption is the public access for related blockchain data. The blockchain information of associated blockchain can be fetched publicly, such as the public blockchain. This is used to verify whether a specific transaction or its related blockchain information is on the paired blockchains or not. With this assumption, the corresponding transaction information can be validated and fetched.

2) *Security Impact Model*: The data for the exchange is recorded in associated blockchains, and blockchains try to keep their immutability. Note that different blockchains have different ability to keep the data immutability since different blockchains have different consensus algorithms, different mining nodes, et al. Some blockchains are relatively easier to be re-branched, which causes data to be rewritten and allows a user to spend its assets twice (double spending).

We introduce a function $reb(n, chainID)$ as in (13) to represent the re-branch possibility of a block when it has n successive blocks in blockchain $chainID$. It is also abbreviated as $reb(n)$ or reb if not conflict. Its value can be calculated from the parameters of the blockchain or its history records.

$$reb(n, chainID) \quad (13)$$

,where n is the number of successive blocks, and $chainID$ is the identification of the blockchain.

With reb , we can compare the re-branch possibility among different blockchains. A weaker blockchain has more possibility to be re-branched than a stronger blockchain, as in (14).

$$reb(n, weak) < reb(n, strong) \quad (14)$$

When different blockchains are involved, the re-branch of one blockchain may impact another blockchain. Suppose there are two cross-chain transactions in an exchange, $tx1$ in blockchain A, and $tx2$ in blockchain B. At time $t1$, $tx1$ is in blockchain A and $tx2$ is in blockchain B, referring to Figure 7. While later (at time $t1+t2$), $tx1$ is still in blockchain A, while $tx2$ is not in blockchain B due to the blockchain re-branch.

Fig. 7. A matched exchange may be mismatched due to the blockchain re-branch. At time $t1$ (the upper one), $tx1$ and $tx2$ exist on blockchain A and B separately, and they match. At $t1+t2$, $tx1$ has no matched conditional transactions as $tx2$ has been re-branched, and then this exchange does not match.

This indicates that the cross-chain exchange is a time-dependent interaction model, as in (15). All cross-chain methods, which depend on associated actions from different blockchains, will be affected. It does matter whether a method checks associated transactions directly or indirectly. As blockchains have possibilities (high or low) to be re-branched, the data for cross-chain exchanges can be changed, and this affects the validity of cross-chain exchanges.

$$tx1 \text{ dep } tx2, \text{ when } t < t_c \text{ and } t_c \text{ is a finite value} \quad (15)$$

, where t means the dependence time.

Take the hash-locking method as an example. A blockchain re-branch may remove the transactions that are used to trigger the hash locking/unlocking actions or the data that record results of the hash-locking from the blockchain. If the associated data on a paired blockchain is not re-branched, the sender on the paired blockchain loses its money.

Due to this impact of a weak blockchain, a user may lose money in a cross-chain exchange. The possibility to lose money $p_{loss_cross-chain}$ is as in (16). It is under the condition that transactions in the target blockchain (suppose blockchain a) are not re-branched while the paired transaction in another blockchain (suppose blockchain b) re-branch.

$$p_{loss_cross-chain} = (1 - reb_a) * reb_b \quad (16)$$

We can compare this to the loss (p_{loss_same}) when the exchange happens in the same blockchain. When the exchange is in the same blockchain, reb_a is equal to reb_b . The corresponding possibility is shown in (17).

$$p_{loss_same} = (1 - reb_a) reb_a \quad (17)$$

Then we can obtain the difference (p_{loss_delta}) of the loss between the cross-chain exchange and the same chain exchange, as in (18).

$$\begin{aligned} p_{loss_delta} &= p_{loss_cross-chain} - p_{loss_same} \\ &= (1 - reb_a)(reb_b - reb_a) \end{aligned} \quad (18)$$

There are several interesting points from (18), which are described as below.

(1) When $reb_b > reb_a$, then the bigger reb_b is, the more possibility for a user in blockchain a to lose money. This indicates that a weaker blockchain (which has more chance to be re-branched) makes users of the stronger blockchain more possible to lose money.

(2) When $reb_b < reb_a$, then the smaller reb_b is, the less possibility for a user in blockchain a to lose money. This indicates that a stronger blockchain makes users of the weaker blockchain less possible to lose money.

(3) When $reb_b = reb_a$, then the possibility to lose money is the same as in the same blockchain.

Thus, we reach a conclusion that a weak blockchain may impact a strong blockchain. It is necessary to discuss how to reduce the impact.

3) *Solution - To Wait Enough Blocks in Paired Blockchain:* It is difficult to ensure the dependent transactions exist in an infinite time. However, we can increase the t_c in (15) to require more successive blocks for a relatively weak blockchain. The notation n_{wb} is used as the number of required successive blocks. It should be large enough to ensure that the possibility to be re-branch is less than a relatively small number δ , as in (19).

$$reb(n_{wb}) \leq \delta \quad (19)$$

For this purpose, the corresponding parameter (n_{wb}) is added to the cross-chain function ('doesTxExpect (tx , $chainID$, ...)'), which judges whether a cross-chain transaction exists or not. This function transforms to 'doesTxExpect (tx , $chainID$, n_{wb})'.

An exchange has paired transactions, which have different n_{wb} s. Each n_{wb} is for the other blockchain. To make receivers have the same condition, receivers acquire their assets when each transaction (block) has its required successive blocks. Suppose there are two users PA and PB to exchange on blockchain A and blockchain B. PA specifies 'doesTxExpect ($tx2$, B , n_{wb})', and PB specifies 'doesTxExpect ($tx1$, A , n'_{wb})'. The receiver of $tx1$ acquires the asset after n_{wb} successive blocks on blockchain B and n'_{wb} successive blocks on blockchain A. The receiver of $tx2$ obtains the asset at the same condition, that is after n_{wb} successive blocks on blockchain B and n'_{wb} successive blocks on blockchain A.

Then the confirmation time ($t_{confirm}$) is the maximum confirmation time of all related blockchains, as in (20). This is not shorter than the confirmation time of each blockchain. It also indicates that fewer steps require less confirmation time, which is another consideration that we want to send only a step for each user.

$$t_{confirm} = \max(t_{confirm_chain1}, t_{confirm_chain2}, \dots) \quad (20)$$

Mining nodes connect to associated blockchains to check whether those conditions match or not. The process includes two steps. (1) Mining nodes check whether associated blockchains have n_{wb} successive blocks after the corresponding conditional transaction or not. (2) If the condition matches, miners try to seal the paired transaction into this blockchain. Meanwhile, the key information of the n_{wb} th block (such as the block header) is also sealed to prove that the n_{wb} th block exists. Honest miners validate this information and decide whether to add new blocks after it or not.

4) *Incentive Mechanism*: The sender should pay miners to seal cross-chain information from other blockchains. The payment should be bigger correspondingly, as miners need to fetch the conditional transaction and check whether it has required successive blocks or not in paired blockchains. Thus, this parameter should not be set to extremely long. There is a balance between security and payment.

In some cases, the fee is relatively low, the associated conditional transaction may not be fetched from the paired blockchain. Users can offer more fees as additional rewards to miners. It requires to introduce a new kind of transaction, whose receiver address is the address of the previous conditional transaction, and the asset value is the additional fee for the miner.

III. IMPLEMENTATION OF TRANSACTION DEPENDENCE BY CONDITIONAL TRANSACTION

In this section, we describe the transaction (the cross-chain conditional transaction) used to express the dependence in exchange, and demonstrate how cross-chain exchange works with conditional transactions.

A. Cross-chain Conditional Transaction

The conditional transaction is a blockchain transaction that specifies the conditions to give the asset to the receiver. The transaction and its condition form the promise-condition pair used in the proposed model.

Compared to the smart contract-based method, the conditional transaction method does not require the deployment of smart contracts on associated blockchains. Its condition is a kind of embedded logic inside a transaction, which is self-expressed. In the cross-chain exchange case, it requires that associated blockchains to support the same condition keywords.

transaction	...	condition	...
-------------	-----	-----------	-----

Fig. 8. Condition field in the transaction

1) *Condition Field in Transaction Structure*: The data structure of a blockchain transaction is extended to include the condition field, as shown in Figure 8.

The condition field helps to express its dependence; the logic of the conditional transaction can be described by algorithm 3. Lines from 3 to 6 specify the conditions. In most cases, line 5 is omitted as it is the default behavior.

Algorithm 3 The transaction with condition fields

```

1: To: <transaction receiver>
2: From :<transaction sender>
3: Condition:
4: if <condition 1> and <condition 2> or < ... > then
5:   Transfer asset to <transaction receiver>
6: end if
7: ...

```

In cross-chain exchange, the required transactions are from different blockchains. To identify different blockchains, the blockchain id (chainID or cid) is used in the condition, as shown in Figure 9.

transaction	...	condition	...	...
			chainID	...

Fig. 9. Cross-chain conditional transaction has blockchain ID fields

The processing of the conditional transactions is to add the required transaction into the exchange checklist. It is shown in Figure 10, and the corresponding algorithm is algorithm 4.

B. How Cross-chain Conditional Transaction Works in Exchange

We use an example to demonstrate how the cross-chain conditional transaction works in the exchange. The scenario is that user 1 who has digital coins on blockchain A wants to exchange with user 2 who has digital coins on blockchain B. Then user 1 plans to send the conditional transaction on blockchain A, $tx1$, which requires a paired transaction from user 2 to user 1 on blockchain B. $tx1$ is shown in algorithm 5. Similarly, user 2 plans to send transaction $tx2$, which transfers the asset from user 2 to user 1 in blockchain B, as shown in algorithm 6.

There are four possible cases with respect to whether users 1 and 2 send their associated transaction or not. Those cases are listed in table II.

We discuss those cases one by one and analyze how they can avoid the partial transfer case that a participant gives its asset without getting its exchanged asset.

In case 1, both two users send their associated transactions. At step 1, user 1 carries out the conditional transaction ($tx1$)

Fig. 10. The process to add the associated transaction list

Algorithm 4 To add required transactions to the check list**Require:**

The cross-chain transaction tx in exchange i ;
 The associated transaction list $requiredTxList$ of exchange i which we are trying to update.

Ensure:

The associated transaction list of exchange i , which has been update by the new required transactions by cross-chain transaction tx .

Ensure:

```

1: if not  $tx$  is cross-chain transaction then
2:   //Not updated
3:   return  $requiredTxList$ 
4: end if
5: for  $rtx$  in required transactions of  $tx$  do
6:   //Only add new required transaction
7:   if not  $rtx$  in  $requiredTxList$  then
8:     Add  $rtx$  to  $requiredTxList$ 
9:   end if
10: end for
11: return  $requiredTxList$ 

```

Algorithm 5 $tx1$ to transfer asset from user 1 to user 2 on blockchain A

```

1: From :user 1
2: To: user 2
3: Condition:
4: if IF doesTxExpect(from: user 2, to: user 1, value:..., chainID:B) then
5:   //Transfer asset to user 2
6: end if
7: ...

```

Algorithm 6 $tx2$ to transfer asset from user 2 to user 1 on blockchain B

```

1: From :user 2
2: To: user 1
3: Condition:
4: if IF doesTxExpect(from: user 1, to: user 2, value:..., chainID:A) then
5:   //Transfer asset to user 1
6: end if
7: ...

```

TABLE II
DIFFERENT CASES FOR EXCHANGE BETWEEN TWO PARTICIPANTS

case	user 1	user 2	comments
1	Tx1	Tx2	successful
2	Tx1	not	send fail
3	not	send	Tx2 fail
4	not	send	N/A fail

and waiting for the required transaction; the digital coins of user 1 has been subtracted (frozen) but does not transfer to user 2. At step2, user 2 carries out the required conditional transaction ($tx2$); and the delivery asset is also frozen without real delivery to user 1.

When both transactions appear, the digital coin is transferred to user 2 on blockchain A, and the asset is delivered to user 1 on blockchain B. All receivers acquire their assets. It is a successful case in which each participant obtains their assets. This is shown in part 'a' of Figure 11.

The transaction sequence does not impact the result, as there is no sequence requirement in the proposed model. If user 2 sends $tx2$ first, it is still a successful case, which is shown in

part 'b' of Figure 11.

In case 2, user 1 sends the conditional transaction $tx1$, while user 2 does not send the paired conditional transaction, $tx2$. This makes the $tx1$ times out, and the frozen assets are given back to user 1. This is shown in part 'c' of Figure 11.

In case 3, this is similar to case 2. In both cases, the assets return to the sender at last, and they avoid the possible partial transfer.

In case 4, there is no associated transaction and then no partial transfer of any asset.

IV. VERIFICATION

In this section, we describe the verification results of the cross-chain exchange model. Those verifications are in two aspects. (1) The typical exchanges between two and three blockchains. It is compared with other cross-chain methods. (2) The more complex exchange. It is achieved by the combination of the conditional logic keyword and the usage of the data fields of a transaction, which is difficult for the other methods to implement.

The blockchains used in the verification are developed according to the one-step model based on the conditional

Fig. 11. Different exchange cases with conditional transaction.

transaction. All blockchains support conditional transactions. PoW is used as the consensus algorithm. It is developed in Java can be run on the environments with JRE. Those blockchains are deployed on virtual machines that run on a workstation, DELL PowerEdge T130, with 4 Intel Xeon CPU E3-1220 v5 @3.00 GHz and 32G memory.

Blockchains monitor their neighbored blockchains and mine the transactions for cross-chain exchanges into its own blockchain. To make the blockchain propagate more quickly, each node of a blockchain connects to all nodes of another blockchain in our verifications.

The compared methods are the hash-locking method and the notary method. The first stands for the methods to lock the asset first and to transfer the asset later. The second represents methods of two-phase commit protocol.

In the hash-locking method, transactions are sent to lock the assets in the first stage and to deliver the hash to get the frozen asset in the second stage. In the later stage, transactions are serialized to ensure the correct transfer of assets. We use a testing script to send those transactions. When transactions to lock the asset are sealed into corresponding blockchains, the script tries to send the second stage transactions in order. If a second stage transaction is sealed into the blockchains, it sends the next one immediately to avoid unnecessary waiting time.

In notary methods, it needs to send transactions in two phases. In the first phase, each blockchain promise to give the asset, which is in a serialized order. In the second phase, each blockchain gives the second commit transaction, which

can be parallelized. We still use a script to perform the test. The script sends first phase transactions in order, in which a transaction is sent when its previous transaction is sealed into the blockchain. Transactions in the second phase are sent when all transactions in the first phase are sent. For the proposed methods, conditional transactions are used for participants. Each one sends a possible transaction under a specific condition.

To measure the difference among those methods, we use the completion time as the measurement to compare different methods. It is calculated from the time when the first transaction is sent to the time when the receivers obtain their assets.

We have verified 30 test rounds for each kind of verification. For a better comparison, the completion time is sorted in ascending order in each diagram.

A. Exchange Between Two Blockchains

In this section, we show the verification results between two blockchains, named blockchains A and B separately. To measure the number of transaction impacts for the exchange, we verify exchanges between 2, 3, and 4 participants.

1) *Exchange between Two Participants:* The exchange is between two participants. One participant uses its asset in a blockchain to exchange another participant's asset in a different blockchain.

The verification results are shown in Figure 12, from which we see the notary and hash-locking methods manifest higher completion time than the proposed method. For the notary method, the average completion time is 24.93 seconds, and the

standard deviation is 4.54 seconds. It waits for two transactions to be sealed into the blockchains (one transaction in each blockchain) in the first stage. For the hash-locking method, it takes almost the same time (average completion time, 25.53 seconds, and the standard deviation, 4.49 seconds). It can be understood that although in the first stage the transactions (to lock the user's asset) can be parallelized; however, transactions to unlock the asset are serialized. Suppose the first user unlocks its asset with transaction $th21$. When $th21$ is sealed into the blockchain, the second user sends the transaction (named $th22$) to get its asset. The exchange complete when $th22$ is sealed. Then the completion time is the summarization of at least three transactions (time of $th21$, $th22$ and time of the first stage, which is at least the time of one transaction). The proposed method takes an average of 9.57 seconds (the standard deviation, 2.60 seconds) to complete, which is the lowest among the three. It only requires to send one round of transactions, and those transactions can be sent in parallel. It has no explicit stage to lock the asset, as the conditional transaction locks the asset automatically. The minimum completion time is not far more than one transaction's sealing time (we send those conditional transactions in parallel in this verification).

Fig. 12. The completion time of cross-chain exchanges between two participants in two blockchains.

B. Exchange among Three and Four Participants

In this verification, there are three or four participants. Each one uses its asset in a blockchain to exchange other participants' assets in different blockchains. As it can show the increasing tendency, we do not increase to more participants. Note we choose the notary method as the comparing method, as the completion time of the hash-locking method manifests the same tendency as that of the notary method in the previous verification.

In 3 participants (3 users) case, for the proposed method, there is 1 conditional transaction in blockchain A and 2 conditional transactions in blockchain B. For notary method, there is 1 transaction in the first stage in blockchain A, and

two transactions in blockchain B. The number of the second commit transactions in corresponding with the number of the transaction in the first stage transaction. In 4 participants case, it is similar, referring to table III.

TABLE III
TRANSACTIONS USED IN THE VERIFICATION (BC IS THE ABBREVIATION OF BLOCKCHAIN AND CT IS THE CONDITIONAL TRANSACTION METHOD)

Method	BC	stage	transaction number 3 users	4 users	comments
Notary	A	1st	1	2	serially
		2nd	1	2	parallel
	B	1st	2	2	serially
		2nd	2	2	parallel
ct	A	N/A	1	2	serially
	B	N/A	2	2	parallel

The verification results are shown in Figure 13, in which the completion time in two participant case is also included.

Fig. 13. Completion time for different users (participants). The notation of 'ct' is the conditional transaction method. '2 users', '3 users' and '4 users' are verifications with 2, 3 and 4 participants separately

From Figure 13, we see that with the increase of the participant number, the completion time of both the conditional transaction method and the notary method increases. The average time is shown in Figure 14. However, the increment of the notary method is more (7.5 and 4.3 seconds of the notary method, comparing to 0.8 and 2.4 seconds of the proposed method). This is due to one more transaction (from the 3rd participant) that needs to be waited for in the first commit stage of the notary method.

C. Exchange among Three Blockchain

In this section, we verify exchange among three blockchains (blockchains A, B, and C). Each blockchain has one participant. The goal is to test the impact when the blockchain number increases. For the proposed method, both blockchains A, B and C send out a conditional transaction. For the notary method, all blockchains execute the two-phase commit.

Figure 15 shows the completion time for the verifications, which are also compared to the case in which 3 participants

Fig. 14. The average completion time for different transaction numbers.

Fig. 15. Completion time of exchanges among 3 participants in 3 blockchains (3bc) and 2 blockchains (2bc). The notation of 'ct' is the results of the conditional transaction methods.

exchange between two blockchains. The completion time of both the proposed method and the notary method increases for 3 blockchain cases. The average time of conditional is 24.57 seconds, and that of the notary method is 69.17 seconds (shown in table IV). It is due to that all associated transactions should be propagated to those three blockchains when blockchain number increases. While the time of notary increases more, as it has to wait for at least 4 serialized transactions (the time of 3 transactions in the first phase and the time of at least 1 transaction in the second phase). The increasing tendency can also be seen in Figure 15.

D. More Flexible Exchange

In this section, we verify that the proposed method has the ability to express transaction dependence flexibly. (1) It adopts conditional keywords to express the different transaction dependence, which does not require to change a smart contract.

TABLE IV
THE AVERAGE COMPLETION TIME COMPARISON BETWEEN THE CASE THAT 3 USERS IN 2 BLOCKCHAINS (2BC) AND THE CASE THAT 3 BLOCKCHAINS (3BC).

Method	3users in 2bc	3users in 3bc
conditional transaction	24.57	9.77
notary	69.17	36.70

(2) And it can make full utilization of the data field of a transaction.

The detailed scenario is that user 1 and user 2 want to exchange their assets on two blockchains (blockchains A and B). User 2 separates its payment into two sub payments at blockchain B. The sum of those two transactions is 12 unit assets. All those transactions are under a specific contract, which should be contained in the corresponding transactions.

In this exchange, one transaction requires more than one transaction in another blockchain. We use the conditional keyword 'AND' to achieve that one transaction in a blockchain requires more than one transaction in another blockchain. Those transactions have a specific agreement (in the data field), which requires to fully use the data fields of a transaction. The proposed method depends on the transactions directly, and it is feasible to achieve this. However, this is difficult for other cross-chain methods. The hash-locking method only provides a proving hash that has no information of the data field. The notary method only uses a receipt for the seconds commit, and it still can not reuse the data field. This exchange is made up of the following transactions.

(*RTx12*) Blockchain A: user 1 transfers 2 assets to user 2 under a specific contract, which is in the data field.

(*RTx211*) Blockchain B: user 2 gives the first sub transaction (2 assets) to user 1 under the same contract, which is specified in the data field.

(*RTx212*) Blockchain B: user 2 gives the second sub transaction (10 assets) to user 1 under a wrong contract, which is in the data field.

(*RTx213*) Blockchain B: user 2 gives the second sub transaction (10 assets) to user 1 under the specific contract, which is in the data field.

We carried out those conditional transactions on their corresponding blockchains. The balance change process is shown in Figure 16. (1) User 1 sends transaction *RTx12*, which requires the two sub transactions from on blockchain B (*RTx211* AND *RTx213*), and those transactions should contain the negotiated exchange contract. This transaction is put to the blockchain at 10:16:30, and the balance of user 1 is decreased 2 assets (from 100 to 98) on blockchain A. (2) User 2 sends transaction *RTx211*, which requires the paired transaction as *RTx12* on blockchain A. This transaction is put to the blockchain at 10:16:34, and the balance of user 2 is decreased from 100 to 98 on blockchain B. (3) User 2 sends transaction *RTx212*, which requires the paired transaction as *RTx12* on blockchain A; while this transaction contains the wrong exchange contract. This transaction is put to the blockchain at 10:16:40, and the balance of user 2 decreases from 98 to 88 on blockchain B. (4) User 2 sends transaction *RTx213*, which requires the paired

Fig. 16. Exchange process of one transaction requires two transaction under specific terms.

transaction as RTx_{12} on blockchain A. This transaction is put to the blockchain at 10:16:47, and the balance of user 2 is decreased from 88 to 78 on blockchain B. RTx_{211} and RTx_{213} matches RTx_{12} , which leads to a successful exchange among user 1 and user 2. RTx_{212} is not under correct exchange contract, whose asset is restored when times out at 10:21:35.

V. RELATED WORK

Special blockchain roles are used in the cross-chain exchange. 'Relay-chain' connects other parachain through four basic roles in the upkeep of a Polkadot network [16]. Transactions are sent first, and the asset transfer is performed when negotiation among the source and destination blockchains is achieved. Chain Fibers classify different roles in blockchain and assume that they are mostly benevolent and cannot all be bribed. Bounded execution windows, participant incentives, and a 'reverse' execution order enable secure payments between parties without shared trust in any system or institution [19].

The cross-chain interaction can also be based on the special blockchain structure. Pegged sidechains enable Bitcoin and other ledger assets to transfer among multiple blockchains, which requires to lock the asset both at the parent chain and side chain. It gives the users access to new and innovative cryptocurrency systems using the assets they already own [17]. Virtualchain is introduced as a logical layer for implementing arbitrary fork-consistent replicated state machines on top of already-running blockchains [20]. MultiChain is an off-the-shelf platform for the creation and deployment of private blockchains, and aims to overcome a key obstacle to the deployment of blockchain technology, by providing the privacy and control required in an easy-to-use package. [21].

The smart contract is also used to implement cross-chain interaction. It is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract, which is a set of programs. It automates the multi-step processes [22]. When it applies to the exchange case, it

can be a specific model [23] or a common model, such as the hash-lock model [24]. It behaves like a public contract, which ensures the exchange. When smart contract-based way applies to the cross-chain scenarios, it requires smart contracts to be pre-deployed on associated blockchains.

Hash-locking based methods [18][24] is based on the smart contract. The asset is locked in a smart contract in the first stage, and in the second stage, assets are withdrawn in pre-defined order using a random-generated hash. The actions in the second stage are serialized [18]. At the same time, it utilizes the hash instead of the transaction itself; the data of the transactions, which may be meaningful to the cross-chain exchange, is not reflected.

To maintain the execution order, two-phase and three-phase commit is proposed. The notary method [19] is a two-phase transaction commit protocol. In the first stage, associated blockchains give their promise (by transactions) to perform an exchange. In the second stage, it executes or aborts the exchange, which depends on the negotiation results among different blockchains. Three-phase commit is similar, which is proposed for heterogeneous blockchain communication [14].

Other works include using the directed graph to analyze the atomic cross-chain swap and indicate the graph should be strongly connected [18][25]. While it only analyzes the transaction relation after the swap has been performed, and the analysis of transaction dependence during the exchange is lacked. Meanwhile, their implementation methods are based on the hash-lock. It requires to pre-deploy associated smart contracts, and the transaction sequence is required.

VI. CONCLUSIONS

In this paper, we address the problem of cross-chain exchange, which needs to solve the partial exchange issue. We propose the one-step model, which is based on transaction dependence to ensure atomicity. Furthermore, we propose to use the conditional transaction to express the transaction dependence. We simulated different blockchains exchange scenarios and performed the corresponding analysis and evaluation. Compared to other cross-chain methods, our proposed method expresses the logic inside the conditional transactions, and it has no extra steps to lock the assets or to perform a second stage commit. Meanwhile, it has no requirement to form special blockchain structures or has special roles in blockchains. Overall, we provide a flexible way to connect different blockchains.

ACKNOWLEDGMENT

REFERENCES

- [1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. bitcoin.org, 2008.
- [2] Wood G. Ethereum: A secure decentralised generalised transaction ledger[AJ]. Ethereum Project Yellow Paper, 2014, 151: 1-32.
- [3] Kalodner H A, Carlsten M, Ellenbogen P, et al. An Empirical Study of Namecoin and Lessons for Decentralized Namespace Design[AC]/WEIS. 2015.
- [4] Underwood S. Blockchain beyond bitcoin[AJ]. Communications of the ACM, 2016, 59(11): 15-17.
- [5] Kakavand H, Kost De Sevres N, Chilton B. The Blockchain Revolution: An Analysis of Regulation and Technology Related to Distributed Ledger Technologies[AJ]. 2017.

- [6] Ahmed S, ten Broek N. Food supply: Blockchain could boost food security[AJ]. *Nature*, 2017, 550(7674): 43.
- [7] Tian F. An agri-food supply chain traceability system for China based on RFID & blockchain technology[AC]//Service Systems and Service Management (ICSSSM), 2016 13th International Conference on. IEEE, 2016: 1-6.
- [8] Kim H M, Laskowski M. Toward an ontology-driven blockchain design for supply-chain provenance[AJ]. *Intelligent Systems in Accounting, Finance and Management*, 2018, 25(1): 18-27.
- [9] Biswas K, Muthukkumarasamy V. Securing smart cities using blockchain technology[AC]//High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on. IEEE, 2016: 1392-1393.
- [10] Sun J, Yan J, Zhang K Z K. Blockchain-based sharing services: What blockchain technology can contribute to smart cities[AJ]. *Financial Innovation*, 2016, 2(1): 26.
- [11] Sharma P K, Moon S Y, Park J H. Block-VN: A distributed blockchain based vehicular network architecture in smart City[AJ]. *Journal of Information Processing Systems*, 2017, 13(1): 84.
- [12] Aniello L, Baldoni R, Gaetani E, et al. A prototype evaluation of a tamper-resistant high performance blockchain-based transaction log for a distributed database[AJ]. 2017.
- [13] Tama B A, Kweka B J, Park Y, et al. A critical review of blockchain and its current applications[AC]//Electrical Engineering and Computer Science (ICECOS), 2017 International Conference on. IEEE, 2017: 109-113.
- [14] Kan L , Wei Y , Hafiz Muhammad A , et al. [IEEE 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) - Lisbon, Portugal (2018.7.16-2018.7.20)] 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) - A Multiple Blockchains Architecture on Inter-Blockchain Communication[C]// 2018:139-145.
- [15] Wang H, Cen Y, Li X. Blockchain Router: A Cross-Chain Communication Protocol[AC]//Proceedings of the 6th International Conference on Informatics, Environment, Energy and Applications. ACM, 2017: 94-97.
- [16] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework[AJ]. 2016.
- [17] Back A, Corallo M, Dashjr L, et al. Enabling blockchain innovations with pegged sidechains[AJ]. URL: <http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains>, 2014.
- [18] Herlihy M. Atomic cross-chain swaps[C]//Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing. ACM, 2018: 245-254.
- [19] Thomas S, Schwartz E. A protocol for interledger payments[AJ]. URL <https://interledger.org/interledger.pdf>, 2015.
- [20] Nelson J, Ali M, Shea R, et al. Extending existing blockchains with virtualchain[AC]//Workshop on Distributed Crypto-currencies and Consensus Ledgers (DCCL'16),(Chicago, IL). 2016.
- [21] Greenspan G. MultiChain Private Blockchain, White Paper[AJ]. URL:<http://www.multichain.com/download/MultiChain-White-Paper.pdf>, 2015.
- [22] Christidis K, Devetsikiotis M. Blockchains and smart contracts for the internet of things[J]. *IEEE Access*, 2016, 4: 2292-2303.
- [23] Kang J, Yu R, Huang X, et al. Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains[J]. *IEEE Transactions on Industrial Informatics*, 2017, 13(6): 3154-3164.
- [24] Buterin V. Chain interoperability[J]. R3 Research Paper, 2016.
- [25] Miraz M, Donald D C. Atomic Cross-chain Swaps: Development, Trajectory and Potential of Non-monetary Digital Token Swap Facilities[J]. *Annals of Emerging Technologies in Computing (AETiC) Vol*, 2019, 3.