Coincer: Decentralised Trustless Platform for Exchanging Decentralised Cryptocurrencies

Michal Zima

Faculty of Informatics, Masaryk University, Brno, Czech Republic
E-mail: xzima1@fi.muni.cz

Abstract. We address the problem of a trustless decentralised exchange of cryptocurrencies. Centralised exchanges are neither trustworthy nor secure. As of 2017, there has been more than 25 million US dollars' worth of cryptocurrencies stolen from (or by) centralised exchanges. With Coincer we allow any two users to exchange their diverse cryptocurrencies directly between them, yet with no need to trust each other. Former approaches either do not do without a server or rely on a trusted issuer of exchangeable tokens. Our approach is to fully eliminate any elements susceptible to becoming a single point of failure. Coincer therefore leverages an efficient anonymous P2P overlay and an atomic protocol for exchanging money across different cryptocurrencies. It is implemented as free software and has been successfully tested with Bitcoin and Litecoin.

1 Introduction

Since the early days of the first clones of the cryptocurrency Bitcoin there has been demand for exchanging one for another. Internet forums and IRC servers were soon superseded by specialised exchange servers. However, the advantage of fast and convenient trading was outweighed by a need to trust the platform and its operator. Unsurprisingly, many of such centralised platforms were successfully cracked and users' funds stolen or they were simply closed down by their operators without returning users' money. As of 2017, losses from these events already exceed 26 million USD [11]. We expect this number to further grow.

In our paper we elaborate on the problem of trustless exchange of cryptocurrencies with a goal to eliminate the aforementioned issue. Present approaches centralise communication or build different structures that still require users' trust in operators of the exchange platform. In contrast, our approach is for a platform to be as decentralised and trustless as possible. We call our platform Coincer. It leverages principles of atomic transactions across distinct cryptocurrencies and deploys a custom P2P network to carry out decentralised market and communication between users.

The remainder of this paper is organised as follows. Section 2 presents previous work related to the problem of exchanging cryptocurrencies in a decentralised manner. Section 3 discusses our design goals and decisions for Coincer, followed by Sect. 4 in which we elaborate in detail on cryptocurrency exchange protocol, the most important component of Coincer. Section 5 describes a way for establishing decentralised markets, followed by a summary of Coincer's limitations in Sect. 6. Section 7 provides final conclusions.

2 Related Work

The first alternative platform for trading cryptocurrencies was Multigateway [10] built on top of Nxt Asset Exchange [7]. It uses NXT[1] tokens to transform arbitrary cryptocurrency to a tradeable asset on this platform. A user has to deposit their cryptocurrency to an escrow formed by 3 servers, each supposedly operated by a different (and independent) operator. These servers cannot manipulate with the deposits on their own, unless at least two of them agree on a transaction. Therefore, to steal funds, two servers need to be cracked. NXT tokens can be easily exchanged as the trades happen on a single "database."

A similar approach has been taken by B&C Exchange [5]. On this platform 15 (independent) servers form the deposit escrow and at least 8 of them are needed for authorisation of a transaction. To use this platform, a user needs "credits" to pay fees for using the platform. All actions and transactions are recorded to a dedicated blockchain, which also eases coordination of the servers.

Unlike the described platforms, a man with a pseudonym "TierNolan" in 2013 designed a protocol based purely on cryptographic and scripting capabilities of cryptocurrencies [2] that should have allowed atomic transactions across two distinct cryptocurrencies to take place [8]. Nonetheless, due to practical issues [3], the atomicity was not actually achieved.

Still, there were several attempts at implementing this protocol [1, 6], but most of them reached only a proof-of-concept stage. So far the most advanced implementation is Mercury [1]. However, it has never progressed to a usable state and its development is now stopped. While it featured a good user interface, it did not offer any decentralised platform—both communication and market establishment were implemented using author's server.

For our work, we redesigned the idea of TierNolan's protocol using today's scripting capabilities of Bitcoin and other cryptocurrencies, and extended it with a decentralised marketplace.

3 Design of Coincer

The design goal is a decentralised platform for trustless exchange of cryptocurrencies. Decentralisation allows to eliminate elements serving as single points of failure in traditional cryptocurrency exchange systems. By moving from a client-server model to a peer-to-peer architecture with inherent redundancy, we allow users to participate in communication and cryptocurrency trading without restraint, even in the presence of failures of parts of the system.

[1] NXT is a cryptocurrency with a design partially different from Bitcoin.

Coincer follows a principle that other user's actions or the system's failures must not cause a user to lose their money, regardless of their state (idle or involved in a trade). This leads to two important features: first, a user never deposits their money anywhere in the system in order to be able to use the system or to place an order. Second, in case of a failure of the system in the middle of a trade or when the other trading party stops cooperating, there is always a fail-safe to retrieve either the original money back or to finish the trade.

Designing a communication protocol for both the P2P layer and Coincer itself, we also take into account several practical aspects. These mainly include extensibility, simplicity (resulting in low overhead) and ease of implementation.

In order to exclude trust from the trading process and not to involve any third party, we rely purely on the programmable nature of cryptocurrencies. We redesign an atomic protocol with modern features of cryptocurrencies, making use of time locks on spending transactions.

The P2P architecture has already been described in our earlier paper [12]. In short, we designed an unstructured P2P network leveraging principles and algorithms from the area of ad-hoc networks. The result is a resilient and anonymous network that enables secure direct communication of any two participants.

The platform would not be complete without a market where users could negotiate prices for their trades. Leveraging the P2P network we design a decentralised marketplace for any cryptocurrency pair in Sect. 5.

4 Trustless Protocol for Exchanging Cryptocurrencies

Purpose of the protocol is to execute an atomic transaction between two distinct cryptocurrency transaction databases. This transaction transfers some amount of money in one cryptocurrency from one user to another user and at the same time some other amount of money in a different cryptocurrency from the other user to the first user. Atomicity in this context means that either both described transfers are fully executed, or none of them is. Figure 1 illustrates this situation on an example of hypothetical cryptocurrencies abbreviated as ABC and XYZ.

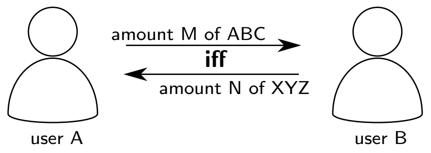


Fig. 1. Atomic transaction of two transfers of different cryptocurrencies.

4.1 Protocol Flow

The protocol comprises two main phases and one "zeroth" preparatory phase. Only the zero phase is interactive—the other two do not involve any communication between the trading parties—all their actions are based solely on states of respective cryptocurrencies' blockchains. This distinction can be easily seen from a diagram in the Fig. 2.

Two phases are needed, because blockchains are independent and we need to assure synchronisation between them. The first phase commits money from both cryptocurrencies into the trade, while the second phase releases money for redemption by their new owners. Until the second phase is started, the trade can be cancelled at any point using rollback transfers. As soon as the second phase begins, the trade can only be finished, but transfer to the new owner is independent of the other user.

In the zero phase, the following data are exchanged[2]:

– public keys;
– scripts that will be used for transactions in the first phase.

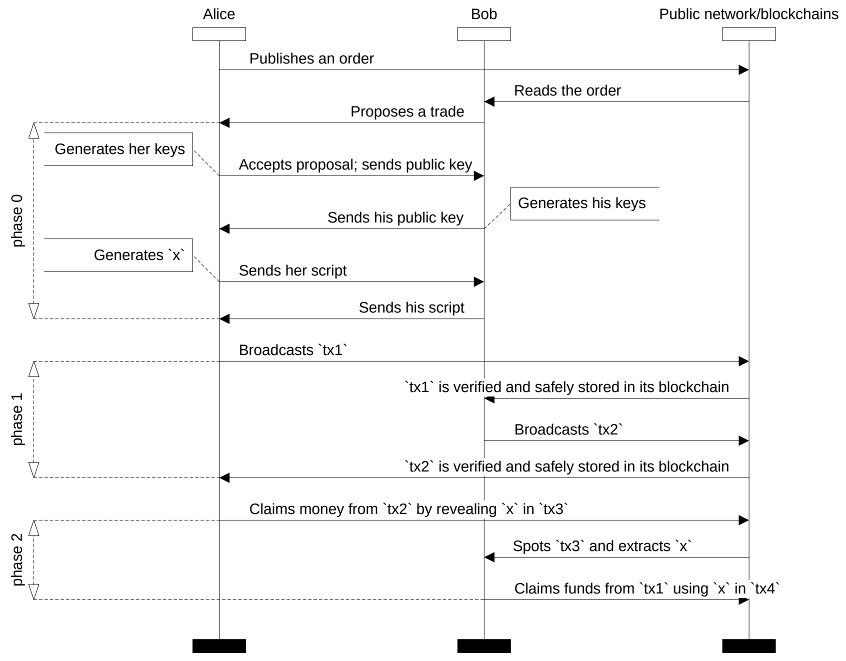
Public keys are needed for proper construction of the script for the second phase, i. e., claiming exchanged funds, and also for managing refund transfers in a case that the trade would be cancelled during the first phase. The scripts themselves are also important, because their knowledge is required in order to claim the funds—the original transactions contain only their hashes and it is up to the consecutive transactions to provide the scripts (this is the principle of P2SH transaction scripts).

Let's assume that Alice starts the protocol. In the first phase, she creates and broadcasts a funds locking transaction *tx1* using a hash of the script that she sent to Bob in the previous phase. As soon as this transaction gets into her cryptocurrency's blockchain and obtains a safe number of confirmations[3], Bob detects this transaction and is able to confirm its correctness. Afterwards, he creates and broadcasts an analogous transaction *tx2* for his cryptocurrency. By the time both transactions are safely stored in their respective blockchains, this phase ends and it guarantees that until time locks (which are specified in the scripts, see details in Sect. 4.2) expire no party can transact using their original money, making it safe to proceed with the second phase.

In the second phase, Alice starts the process of claiming exchanged cryptocurrency funds. She creates a transaction *tx3* that embeds Bob's original script (which he transferred to Alice in phase zero) and its required parameters in order to satisfy its spending conditions. As soon as Alice broadcasts this transaction and Bob detects it, he proceeds with his claiming transaction *tx4*, reusing unlocking information from Alice's transaction. Since there is no reason for him to wait for Alice's transaction to get into the blockchain and confirmed, he broadcasts his transaction without any delay. When both transactions *tx3* and *tx4* are safely saved in their blockchains, the second phase formally ends.

[2] Note that we expect that both parties already settled on cryptocurrency amounts and an exchange rate for their trade. Therefore, these two pieces of information are not included in our description of the protocol.

[3] "Number of confirmations" is a way of expressing depth of a transaction within the chain of blocks. E. g., a transaction in the top block has 1 confirmation, a transaction in a block right below the top block has 2 confirmations and so on.



Alice
Bob
Public network/blockchains
Publishes an order
Reads the order
Proposes a trade
Generates her keys
Accepts proposal; sends public key
Generates his keys
Sends his public key
Generates `x`
Sends her script
Sends his script
phase 0
Broadcasts `tx1`
`tx1` is verified and safely stored in its blockchain
Broadcasts `tx2`
`tx2` is verified and safely stored in its blockchain
phase 1
Claims money from `tx2` by revealing `x` in `tx3`
Spots `tx3` and extracts `x`
Claims funds from `tx1` using `x` in `tx4`
phase 2

Fig. 2. Diagram of the protocol flow.

4.2 Protocol Scripts

The protocol is built upon a cryptocurrency script that allows locking funds until a "password" is provided or until a time lock expires. The script for a transaction sent by Alice is detailed in the Fig. 3, nonetheless, for Bob the script looks the same, only with public keys swapped and with a different time lock.

```
OP_IF
  OP_HASH160
  <RIPEMD160(SHA256(x))> OP_EQUALVERIFY
  <Bob's public key>
OP_ELSE
  <current timestamp + 24 hours>
  OP_CHECKLOCKTIMEVERIFY OP_DROP
  <Alice's public key>
OP_ENDIF
OP_CHECKSIG
```

Fig. 3. Cryptocurrency script for the atomic exchange protocol.

Value x is a locking secret which is known only to the party that sends their transaction first. Let's suppose this is Alice. Then Bob knows only the double hash of x, and also sets his timeout value in `OP_ELSE` branch to a lower value—usually a half, i. e., 12 hours in this example.

4.3 Semantics of the Script

Spender of this transaction chooses a branch to execute. Bob spends Alice's transaction (i. e., claiming his money acquired from the trade) by executing the first branch and providing the value of x and a signature. The signature ensures that no other user can claim the money when the value of x becomes publicly known from the blockchain. For Alice, who executes the second branch, the main requirement is to set time lock of the spending transaction to at least the value given here. This ensures that her transaction will not be processed earlier than it should, giving Bob enough time to finalise the trade[4]. Again, Alice has to provide a signature to prove that she is the rightful owner of the funds.

4.4 Protocol Fail-safe Points

In any moment, the trade can be interrupted and either fully reverted or fully finished, depending on its momentary state. We examine each protocol block:

- the zero phase: trivial—nothing needs to be done;
- the first phase after the first party sends their transaction: the second party does not send their transaction and the first party waits for their timeout to expire in order to reclaim their funds;
- the first phase after both parties send their transactions: they both wait for their timeouts to expire and then reclaim their funds;
- the second phase after the first party redeems the exchanged funds: as the value of x is revealed and the second party's funds are already claimed, the second party can only redeem their exchanged funds.

4.5 Avoiding Race Conditions

While the protocol is designed as atomic, since it makes use of timeouts, there are certain unavoidable edge cases possibly leading to race conditions. For instance, if Alice waited with proceeding into the phase two till near the timeout of Bob's transaction, she risks a race condition in which Bob may attempt to proceed into the rollback phase. If Bob won the race condition, he would hold his original money plus he would know the value of x (from Alice's attempt to claim Bob's money) which would enable him to also claim Alice's money.

Caution is needed not only with regard to absolute time, but also with regard to specifics of cryptocurrencies. Firstly, inclusion of a transaction into a block is not usually instant so some time should be accounted for possible delays. Secondly, due to their decentralised nature, cryptocurrencies maintain a tolerance for time shifts of individual nodes in their networks and of individual blocks, too.

[4] A reader of this paper who is not very familiar with cryptocurrency script languages may wonder about the use of `OP_DROP` operation (which removes a top element from the stack). Its use relates to the definition of `OP_CHECKLOCKTIMEVERIFY`, which leaves an extra item on the stack to be backward compatible with older clients that still interpret this operation as `OP_NOP2` (i. e., "no operation") [9].

For example, Bitcoin allows time of a block to be shifted by up to two hours. A transaction with a time lock could be in this way included into a block up to two hours earlier than the time lock would in fact expire. To lessen this issue, Bitcoin compares time locks not against block's time, but against a median time of last 11 blocks [4]. Nonetheless, a possibility of a time shift attack still exists.

Thence, users should proceed with the protocol without unnecessary delays. If they are left with an insufficiently short time frame (e. g., due to a delay in inclusion of a transaction into the blockchain), a safer rollback is preferred.

4.6 Protocol Atomicity

In this section we show that the presented protocol is atomic and complete (i. e., does not allow any other than specified actions) by having these two properties:

Liveness. There always eventually exists a step each party can do either to make progress within the protocol, or to revert its effects.

Safety. The protocol never deadlocks, and neither party can acquire both parties' money from the trade.

Liveness. For liveness, we want to disprove that one of the parties can perform such an action that the other party cannot perform any action in a finite time. Therefore, we will do an analysis for each step of the protocol. We assume that Alice starts the protocol.

As the very first step, during the zero phase, Alice and Bob generate and exchange their public keys. If one of them does not send their public key, the protocol stops as the other party can trivially terminate it. If they send something different than their public keys, they will lose their money sent into the exchange[5]. Alice then generates a cryptographically secure random number x and creates a script as shown in Fig. 3 and sends it to Bob. If she does not generate such a number, she may lose her money if Bob can figure its value out. If she does not send Bob a script he expects, he trivially terminates the protocol when he verifies its content. The same applies if Alice does not send anything.

Afterwards, Bob extracts the hash of x and creates an analogical script using a lower timeout value and sends it to Alice. Similarly to the previous case, if he does not send Alice the expected script or does not send anything at all, she trivially terminates the protocol.

Now, in the first phase, Alice broadcasts her transaction *tx1* which consists of a hash of the script she previously sent to Bob. Bob knows the hash and monitors the blockchain for a transaction containing it. If Alice does not send the transaction at all[6] or sends a different amount of money, Bob can trivially

[5] Such an action violates the protocol, however, since this does not cause any harm to the other party, it does not break the given property.

[6] If Alice alters the original script, its hash will differ and the transaction will not get recognised by Bob, therefore seeming as if it was not sent at all.

terminate the protocol. In such a case, Alice would need to wait until the timeout expires so that she can claim her money back.

After Alice's transaction is accepted into the blockchain and receives a safe number of confirmations[7], Bob broadcasts his transaction *tx2* which consists analogically of a hash of the script he previously sent to Alice. If the transaction is different than Alice expects or if Bob does not send it at all, Alice waits until the timeout of her transaction *tx1* expires and terminates the protocol by claiming her money back. Bob would need to do the same with his transaction.

As soon as Bob's transaction *tx2* is safely included in the blockchain, Alice starts the second phase. She claims Bob's money in *tx3* using Bob's script (from the zeroth phase) and the value of x. When she broadcasts this transaction and Bob notices it, he constructs an analogical transaction *tx4* with Alice's script and the now-public value of x and broadcasts it. The protocol is then finished.

If Alice decides not to carry on with the second phase, both Bob and Alice have to wait for timeouts of their transactions to expire and in time claim their funds back. If only Bob decides not to continue with the trade, he is likely to lose money as soon as the timeout of Alice's transaction *tx1* expires. At this point there is nothing he could do to affect Alice and cause her harm as she proceeded with the second phase and already claimed money from Bob's transaction *tx2*.

Safety. Safety property says that the protocol should be deadlock-free and that no party can steal money of the other party. We already know that the protocol is deadlock-free from liveness. Therefore, it only needs to be shown that no trading party can lose their money while conforming to the protocol.

Money is sent to the trade only via transactions with a hash of script detailed in Fig. 3. Spending money from such a transaction requires supplying such data that its script evaluates to *true*. This specific spending transaction requires the original script, that was hashed, which will execute exactly one of its branches and will always check a signature.

The first branch performs an equality verification of the given hash with a hashed value of x that was provided in the spending transaction, and puts a requirement for a signature made by the party that is supposed to receive this transaction's money, which is done by specifying a proper public key. The knowledge of the value of x alone is ergo insufficient for claiming the money.

The second branch is controlled solely by the original sender. It guarantees that if the first branch is not executed within the time limit specified in the second branch, the second branch can be executed, but not earlier—executions before the specified time would fail and result into an invalid spending transaction. Therefore, it can only be used within the "rollback phase" of the protocol. If the money remains unclaimed (i. e., the trade is interrupted), this branch ensures that the original owners will receive their money back. To execute this branch the original owner needs to supply a signature with a private key corresponding to the specified public key. Therefore, no one else can execute this branch.

[7] Number of confirmations that is considered safe is dependent on particular cryptocurrency. For Bitcoin this is usually 2–6 (based on context and amount).

In the beginning of the protocol, only Alice knows the value of x, but she cannot use it until Bob broadcasts his transaction in the first phase which happens iff Alice first broadcasts hers. While she can claim money from Bob's transaction, she would have to wait until her transaction's timeout expires—with Bob not following the protocol and not claiming money from Alice's transaction using the value of x. Bob has no way to claim Alice's money and to keep his money at the same time since his money is already claimed by Alice by the time he learns the value of x. Due to timeouts, race condition cannot happen.

Therefore, we have shown atomicity of the protocol.

5 Decentralised Markets

Before a trade can even take place, two parties have to settle on terms of the trade first. Our goal is a resilient platform where each node is independent of the rest of the network and which is resistant to withholding information about asks and bids of other users from reaching a particular node. In order not to introduce a single point of failure, we avoid solutions involving servers or prominent nodes. Market information should be distributed among nodes all over the network.

Placing an order. Publishing an ask or a bid is performed by broadcasting a message with information about amounts and cryptocurrencies being sold/bought. Since the messages are broadcast, every node learns about every order. Nodes maintain multiple connections to the network, hence an incidental node that does not forward an order does not prevent its neighbours from receiving it.

Exploring a market. Nodes primarily gather data about markets from orders received from other nodes. Nevertheless, unless a node has been connected to the network for a certain time span, its view of the market state might be only partial. To retrieve missing orders, nodes ask their neighbours for orders related to a specific cryptocurrency pair—either when they connect to the network or when they need the orders in order to start a trade.

Market maintenance. To keep a list of orders useful, stale and out-of-date orders should be purged. A user may cancel their order explicitly. Otherwise, all orders are bound to expiration after 1 hour. If an order is still valid after 1 hour since its initial publishing, the user needs to rebroadcast it.

Identification of cryptocurrencies. When an order is sent, it needs to distinguish individual cryptocurrencies. Since verbal descriptors might be sometimes ambiguous or even change (e. g., a cryptocurrency Dash used to be called Darkcoin, but even before that it was named XCoin), we propose as a cryptocurrency identifier a hash of the first block within the cryptocurrency that differentiates it from the original cryptocurrency (if there is any—as in a case of a fork), which is most times the very first block in the chain. Each cryptocurrency is hence uniquely and consistently identified within the platform. Applications can then do a mapping and present users with the cryptocurrencies' common names.

6 Limitations

The most notable limitation of Coincer, when compared to centralised exchanges, is non-real-time nature of trades. A single trade can take from tens of minutes up to several hours, because for every trade a user has to perform a money deposit, while a centralised exchange requires only one deposit for arbitrary number of trades that may finalise nearly instantly.

Besides that, from a technical nature of various cryptocurrencies, it is possible to find two cryptocurrencies with a different set of supported cryptographic operations. If two cryptocurrencies do not support the same operations that are needed by the atomic protocol, it is not possible to exchange them directly. Instead, an intermediary cryptocurrency must be used.

7 Conclusion

All different kinds of cryptocurrencies are nowadays exchanged by people via centralised services. However, many such services have been either cracked or intentionally shut down, rendering more than 21 million USD in losses to users as of 2017 [11]. To prevent such events, several platforms try to decentralise exchanging services.

Coincer is our fully decentralised solution for exchanging cryptocurrencies. Its trading layer is built using an atomic two-party cryptocurrency protocol, thence removing any intermediaries from the process and making the exchange safe and trustless. Communication and pairing of bids and asks are fully carried over a P2P network, making Coincer a serverless platform without any single point of failure.

References

1. Bell, M.: Mercury, `https://github.com/mappum/mercury`
2. Bitcoin Wiki: Script, `https://en.bitcoin.it/wiki/Script`
3. Bitcoin Wiki: Transaction Malleability, `https://en.bitcoin.it/wiki/Transaction_Malleability`
4. Kerin, T., Friedenbach, M.: BIP 113: Median time-past as endpoint for lock-time calculations (Aug 2015), `https://github.com/bitcoin/bips/blob/master/bip-0113.mediawiki`
5. Lee, J.: Blocks & Chains Decentralized Exchange, `https://bcexchange.org/assets/Blocks_&_Chains_Decentralized_Exchange.pdf`
6. Nicoll, R.: Cate, `https://github.com/rnicoll/cate`
7. Nxt Wiki: Asset Exchange, `https://nxtwiki.org/wiki/Asset_Exchange`
8. TierNolan: Re: Alt chains and atomic transfers (May 2013), `https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949`
9. Todd, P.: BIP 65: OP_CHECKLOCKTIMEVERIFY (Oct 2014), `https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki`
10. VanBreuk: Multigateway: Service Documentation v0.2 (Aug 2014), `https://multigateway.org/downloads/Multigateway_docs.pdf`
11. Zima, M.: History of Cracks, `https://www.coincer.org/history-of-cracks/`
12. Zima, M., Hladká, E.: Cryptography enhanced ad-hoc approach to P2P overlays. pp. 517–522. IEEE (Jul 2016), `http://ieeexplore.ieee.org/document/7568378/`