Towards A Broadcast Time-Lock Based Token Exchange Protocol

Fadi Barbàra[1]✉, Nadir Murru[2], and Claudio Schifanella[1]

[1] Department of Comuputer Science, University of Turin, Italy
{fadi.barbara,claudio.schifanella}@unito.it
[2] Department of Mathematics, University of Trento nadir.murru@unitn.it

Abstract. Many proposals for token exchange mechanisms between multiple parties have centralization points. This prevents a completely trustless and secure exchange between those parties. The main issue lies in the fact that communications in projects using a blockchain are asynchronous: classical result asserts that in an asynchronous system a secure exchange of secrets is impossible, unless there is a trusted third party. In this paper, we propose our preliminary results in the creation of our Broadcast Time-Lock Exchange (BTLE) protocol. The core of BTLE is the introduction of synchronicity in communications through the use of time-lock puzzles. This makes it possible to exchange secrets between two parties while eliminating the need for a trusted third party.

1 Introduction

Since the introduction of Bitcoin, a plethora of blockchain-based digital currencies have being created. These systems are very different in terms of design and purpose. Initially the projects tried to solve some of the problems in Bitcoin [Poe15], such as increasing the number of transactions per second or creating a decentralized consensus method that would use less energy resources. This need has increased over the years, in particular with the rise of DeFi [WPG$^+$21]. Because it is unlikely that there will emerge a token capable of solving all problems the different designs encounter, blockchain interoperability (also called *cross-chain communication*) is an important research problem.

The methods of cross chain communication to date can be divided into two macro categories [But16]: centralized and decentralized. Centralized methods are methods in which participants send their funds to a central institution (e.g. an online exchange) that takes care of distributing them between the parties at a later date. The advantages of centralized methods are ease of implementation, easy of use and speed. Unfortunately, however, the disadvantages are greater: a central party can steal funds, deny access to funds and in general central parties cannot guarantee the privacy of users.

On the other hand, decentralized exchanges suffer from opposite problems. Although they do not depend on central entities that could jeopardize the safety of the participants, today these exchanges are difficult, slow to implement and require parties to be online for the whole duration of the exchange. An analysis of the different methods for Cross-Chain Communication is made in Zamyatin et al. [ZAZ+18]. As highlighted by the authors, all decentralized exchanges described in the literature involve only two parties. It is therefore assumed that the parties know each other beforehand: how to choose a partner is always left out of the protocol description. This makes it difficult to implement decentralized trades that mirror those of markets where a participant wants to exchange his tokens at an advantageous price, but is not interested in the identity of the partner. Solution for this problems are automatic market makers [XVP+21], non custodial exchanges with centralized order-books (also called *continuous* order books) and matching system [BMRS18]. Other proposals are decentralized order-books [m5221], but it's still unclear whether they can handle the liquidity of traditional centralized markets.

The impossibility result of Fair Exchange [ASW98] states that in an asynchronous setting (as it is the one in blockchains) it is impossible to have secure fair exchange without a third party. Instead of relying on central parties or coordinators and to keep the protocol peer-to-peer, we decided to rely on synchronous communications. The idea is not new, and generally decentralized exchanges rely on Hash Time Locked Contracts (HTLCs), i.e. smart contracts that can be opened only by knowing a secret (in the form of an hash pre-image) after a certain time. The invention of these contracts is credited to Tier Nolan who explained its work on Bitcoin Forum [Nol] and it has been widely studied, for example in [Her18,MD19] in the context of atomic swaps. Interestingly, HTLCs are also the base of payment channels such as Lightning Network [Rus19].

From the studies on Lightning Network, we know that one of the problems of HTLCs is that they require that both parties are constantly online for the protocol to be safe [NFSD20]. Also with HTLCs it is not possible to have more open channels. Only recently Malavolta et al. [MMS+19] have created a way to create multiple channels of exchange of funds from a single node, but these require that the available capital is divided between the various channels. It is not therefore possible to create general proposals of exchange of funds, as when orders are generated in an exchange market, but it is necessary to decide beforehand with whom to communicate/exchange funds.

To solve these problems, we decided to obtain synchronous communication between blockchains in a different way, namely by using time-lock puzzles [RSW96]. Using a time-lock puzzle we can divide the exchange into two phases. In the first phase we use a time-lock puzzle to create an exchange proposal and collect the availability of different participants. In the second phase we make the actual exchange with the winner of the first phase. Here we could use an HTLC, since the parts have been reduced to two, but we decided to use the time-lock puzzle again to leave the possibility for the parts not to be constantly online. This gives the possibility to have a secure exchange even in case of network interruptions.

The major contributions of our paper are:

– We present our preliminary results on Broadcast Time-Lock Exchange (BTLE): a new protocol for a broadcast (or multi-party) atomic exchange of token in a cross-chain scenario
– For optimization reasons, we investigated possible alternatives to the famous RSA-based time-lock puzzles [RSW96]. In details, we propose a new time-lock puzzle based on the conic-based cryptosystem presented in [BM16], which can be of independent interest.

The structure of the paper is the following: in Section 2 we present the relevant literature on time release cryptography and cross chain communication, then in Section 3 we present briefly the classic RSA-based time-lock puzzle of Rivest et al. [RSW96]. We use that to present the details of the inner working of our new conic-based RSA-like time-lock puzzle. On Section 4 we present BTLE. Finally in Section 5 we conclude.

2 Related works

We use time-lock puzzles in a cross-chain communication protocol. For this reason it is useful to analyze the literature in two separate way. The first subsection analyzes the proposed time-related cryptosystems, while the second analyzes different cross-chain protocols.

2.1 Time Release Cryptography

On this topic there is a detailed survey by Jaques et al. [JMR20]. In this subsection we present only the fundamental results related to the BTLE protocol.

Timed primitives came up in several contexts. We can distinguish between a pre-blokchain phase and a post-blockchain phase. The first generation, pre-blockchain, includes "time capsules" for key escrowing as in Bellare et al. [BG96], time-based cryptographic secrets as Rivest et al.in [RSW96] and contract signing as in Boneh et al. [BN00]. Of those, only the last two protocols are secure against parallel processing, i.e. they use what has been defined as *inherently secure* function [BBBF18a].

After the introduction of Bitcoin [Nak08], time based cryptography had a new wave of research. In particular, the post-blockchain study of time-based cryptographic protocols is focused on *verifiable delay functions* (VDF) and *time-lock puzzles* (TLP). The difference is that VDFs are time-lock puzzles whose solution is publicly verifiable without the need to solve the puzzle [BBBF18a].

The majority of the newer studies have focused on VDFs. Some of them proposed new protocols, such as [MT19], while others extended the TLP in [RSW96] making it a VDF. The works of the latter type are that of Wesolowski [Wes20] and that of Pietrzak [Pie18]. These works are compared in [BBBF18b]. Interestingly, the new wave of research on time based cryptography has generally left aside the traditional time-lock puzzles. Still, because we do not need any outside verification of the result (our setting has an implicit verification: if the solution is right, then the user can retrieve the funds, otherwise it can not), we use the more heavily studied construct of time-lock puzzle.

2.2 Cross Chain Communication

In this paper, we focus on decentralized cross-chain exchanges, because centralized exchange are out of scope.

The first report on the interoperability issue is that of Buterin [But16], in which describes the main methods of interoperability at the time. In many cases the realization of an interoperable system was leveraged to create a system that increased the scalability in terms of transactions of the blockchain in question. In fact, interoperability gave the possibility of rebalancing the loads between two blockchain and then divide the work between the two. In this sense we talk about sidechains, introduced for the first time in [BCD$^+$14]. Several projects were born that aim to create "a blockchain of blockchains", such as Cosmos [KB], Polkadot [Woo16] and Plasma [Poo17]. In both projects, the idea is to create a hierarchy of blockchains and each exchange of funds is approved through the blockchain at the head of all others. In these projects the exchange is neither atomic nor peer-to-peer.

To date, the peer-to-peer exchange methods are all based on the concept of Hash Time Lock Contract (HTLC), whose inventor is considered Nolan [Nol] and are analyzed for example by Herlihy [Her18] and by Miraz and Donald [MD19]. An HTLC is a contract that use hash-based and time-based locks to lock and unlock funds. Participants in this kind of contract have to manually redeem funds by generating cryptographic proof of payment before a certain date in order proceed with the protocol. This requires parties to stay online, which is difficult for power constrained devices of in places where continuos internet connectivity can't be assumed.

This type of contract is not implementable on any blockchain because it needs a scriptable blockchain and requires that different blockchain have the same hash function. These are not common requirements: for example the blockchain protocol in Monero does not have the concept of smart contract and can not implement HTLC, and it wouldn't be possible to make a HTLC between Tezos (which uses the Blake2b function as its principal hash function) and Bitcoin (which uses the SHA256 function to make hashing).

For a more detailed analysis see e.g. the work of Zamyatin et al. [ZAZ$^+$18].

3 Two time-lock puzzles

In this section we do a brief digression on how to measure time in a time-lock puzzle and then we present both the classic time-lock puzzle presented by Rivest et al. which we call it RSW-TL, and we present the new time-lock puzzle based on the RSA-like system in [BM16], which we call it BM-TL. Both systems use functions that are believed to be sequential, meaning they are not parallelizable. Therefore, given a particular type of CPU, there is no advantage in having more of them: all computations are necessarily done on only one core of the CPU. These time-lock puzzle can be classified as a CPU-bound puzzle with a timing function and an implicit verification [ACDP20].

On time Since computation is sequential, it's possible to predict the time to solve the puzzle. Given T the time such that A wants to keep B busy, and S the number of squaring per unit of time (either done by the RSW-TL method or the BM-TL method, see below), then $t = TS$ is the number of squaring needed. Obviously S depends mainly on the processor used. As of now, we are doing the necessary tests to see which of the two proposed methods is the best for the BTLE protocol, i.e. has the least variation based on the CPU.

RSA-TL In [RSW96], the authors proposed a simple yet effective time-lock puzzle exploiting repeated squares. A time-lock puzzle is used to encrypt a secret sk (which could be, e.g, the key of a symmetric cryptosystem) so that it could be decrypted only after a fixed amount of time T.

In particular, said A the entity that creates the time-lock puzzle, A encrypts sk as

$$c \equiv sk + a^{2^t} \pmod{n}$$

where $n = pq$, with p and q prime numbers, $0 < a < n$ a random number and t a positive number computed as before. The value of c can be efficiently computed if p and q are known. Indeed, in this case, one can compute $e \equiv 2^t \pmod{\varphi(n)}$ and then $a^e \pmod{n}$ exploiting Euler's totient theorem. The entity A sends (n, a, t, c) to the entity B that has to recover sk.

The entity B can not perform $a^e \pmod{n}$ efficiently because it doesn't know p and q, nor can it derive them from n. In fact, multiplication of "big" primes is the same trapdoor function used by the RSA cryptosystem. Therefore it has to perform t squarings, since the computation of a^{2^t} is believed not to be parallelizable.

BM-TL The idea of Rivest et al. for creating time-lock puzzles can be easily adapted using different products for performing the powers. In [BM16], the authors developed an RSA–like cryptosystem based on a particular parametrization of the Pell conic. We recall here some details. The Pell conic is defined as

$$\mathcal{H} = \{(x, y) \in \mathbb{F} \times \mathbb{F} : x^2 - Dy^2 = 1\},$$

where $\mathbb{F}$ is a field and D square-free, meaning that there is no square in its prime decomposition. It is well known that $(\mathcal{H}, \otimes)$ is a group, where $\otimes$ is the Brahmagupta product defined by

$$(x_1, y_1) \otimes (x_2, y_2) = (x_1x_2 + y_1y_2D, x_1y_2 + x_2y_1).$$

A set of parameters can be found using the line $y = \frac{1}{m}(x + 1)$, yielding to the group $(P, \odot)$ isomorphic to $(\mathcal{H}, \otimes)$, where

$$P = \mathbb{F} \cup \{\alpha\}, \quad a \odot b = \begin{cases} \frac{ab+D}{a+b}, & \text{if } a + b \neq 0 \\ \alpha, & \text{if } a + b = 0 \end{cases}$$

with $\alpha \notin \mathbb{F}$ the point at infinity. When $\mathbb{F} = \mathbb{Z}_p$, p prime, we have that

$$a^{\odot p+1} \equiv 1 \pmod{p}$$

for every $a \in P$, where the powers are evaluated with respect to the product $\odot$. The Pell conic $\mathcal{H}$ and the set of parameters P can be also constructed over rings and considering $P = \mathbb{Z}_n \cup \alpha$, with $n = pq$, p and q primes, we have an analogue of the Euler's totient theorem:

$$a^{\odot \Psi(n)} \equiv 1 \pmod{n}, \quad \forall a \in \mathbb{Z}_n^*,$$

where $\Psi(n) = (p+1)(q+1)$. Finally, we recall that the powers $z^{\odot n}$ can be evaluated by means of the Rédei rational functions:

$$z^{\odot n} = Q_n(D, z)$$

where $Q_n(D, z)$ is the n–th Rédei rational function defined by

$$Q_n(D, z) = \frac{A_n(D, z)}{B_n(D, z)}, \quad (z + \sqrt{D})^n = A_n(D, z) + B_n(D, z)\sqrt{D}$$

and the polynomials $A_n(D, z)$, $B_n(D, z)$ can be evaluated by

$$\begin{pmatrix} z & D \\ 1 & z \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} A_n(D, z) \\ B_n(D, z) \end{pmatrix}.$$

For proofs and further details, see [BM16].
Thus, we can construct a time-lock puzzle following the idea of Rivest et al. [RSW96], but exploiting the product $\odot$. In this case, the secret *sk* is encrypted by

$$c \equiv sk + a^{\odot 2^t} \pmod{n}.$$

Knowing the factorization of n, one can efficiently compute $a^{\odot 2^t}$ evaluating first $e \equiv 2^t \pmod{\Psi(n)}$ and then $a^e \pmod{n}$. Without knowing the factorization of n, one must perform t squarings with respect to the product $\odot$.

4 The Broadcast Time-Lock Exchange Protocol

We now explain how it is possible to use time-lock puzzles to create an alternative to order books and AMMs in decentralized token exchanges. We assume a decentralized platform where participants want to exchange tokens. We assume each participant has a client which follows the defined blockchain protocol. Because the setting is decentralized, it makes no sense speaking of synchronization between these clients. For this reason each participant has its own *view* of the current state of the decentralized market. Without loss of generality, we can say that the participant that initiates the process is selling its token in exchange of (or *to buy*) the other.

The Broadcast Time-Lock Exchange (BTLE) protocol needs two classes of participants. The first class is the *initiator*: this kind of participant is the one who initiates the exchange by proposing the deal, i.e. the selling of its token. Using the terminology of traditional centralized markets, this participant corresponds to a *market taker*. The other class of participants is that of *exchangers*: these participants are possible buyers interested in an equivalent deal but who do not want to start it. The analogue in traditional markets is that of *market makers*.

In this protocol there is a single initiator which we call Alice for simplicity and which we denote by A, and many possible exchangers. Since the exchangers represent Alice's partner, following the cryptography tradition we will call them all "Bobs". Moreover, supposing that they are indexed and that they are in finite number d, we will say that the potential exchangers are $\{B_1, B_2, \dots, B_d\}$. Recall that we are in a decentralized environment and therefore there cannot be a temporal-based ordering of possible buyers: asynchronous systems imply the absence of a shared clocks.

Furthermore, we suppose the set $\{B_1, B_2, \dots, B_d\}$ is completely determined by the view of A: since we are in an asynchronous environment, it is possible that Alice's view of potential buyers is not synchronized. This means that in some other nodes there are other potential exchangers or that some have withdrawn. In the protocol description we will see why neither of these two cases is a problem. Finally we assume that there are d secure channels of communication between A and each one of $\{B_1, B_2, \dots, B_d\}$. Finally, we assume $\{B_1, B_2, \dots, B_d\}$ compete at the same price level, as in traditional order books.

The BTLE protocol is divided into two rounds. In the first round Alice (the initiator) chooses from among the potential exchangers the one with whom the real token exchange will take place. In the second round the actual exchange takes place. In both stages a time-lock puzzle is used.

4.1 Choosing an Exchanger

In this round Alice has to choose the exchanger among $\{B_1, B_2, \dots, B_d\} \leftarrow GetExchangersList()$. It will follow the routine explained in Figure 4.1. In this subsection and in the following one we treat the time-lock puzzle TLP as a blackbox which takes two inputs and then outputs a cryptographic puzzle. The solution of the puzzle is the cleartext itself: that's why we chose time lock puzzles with implicit verification. Also, we see the modularity of BTLE that can support multiple types of time-lock puzzles, either from those presented in Section 3 or even different ones.

As seen in the Figure 4.1, A generates a random message for each participant in $\{B_1, B_2, \dots, B_d\}$ and associates this message to the intended receiver. The inputs of the time-lock puzzle are the message and a time in seconds. The output is a tuple that represent the cryptographic puzzle (See Section 3). A performs this subroutine for all $B_i, i = 1, \dots, d$. Then it sends all the puzzles and waits for a solution. When A receives the first solution (i.e. the cleartext of the random message) from some B_j, it checks that is a valid message, i.e. the cleartext is equal to the message associated with B_j. If that is the case, A accepts the solution and B_j is the winner: A will proceed to communicate only with B_j and discards all other solutions. If the message isn't valid, A waits for another solution.

Note that in a time-lock puzzle $TB_i = TLP(rand_i, time)$, the random message is different for each B_i, but *time* is equal for all participants: all potential exchangers must have the same chance of being able to find the solution at the

Algorithm 1 Round 1: Choosing the Exchanger

```
1: ${B_1, B_2, \dots, B_d} \leftarrow GetExchangersList()$
2: for $i = 1, \dots, d$ do
3:     $A$ generates $rand_i$
4:     {map $rand_i \to B_i$}
5:     $A$ computes the time-lock puzzle $TB_i = TLP(rand_i, time)$ of $B_i$
6: end for
7: for $i = 1, \dots, d$ do
8:     $A$ sends $TB_i$ to $B_i$
9: end for
10: accepted_solution=0
11: while accepted_solution==0 do
12:     wait_for_solution
13:     if verify_solution($sol$)==1 then
14:         accepted_solution=1
15:         winner_sol=$sol$
16:     end if
17: end while
18: return map($sol$)
```

same time. The randomness of the winner is determined by unpredictable factors such as network latency or puzzle real solving time.

From this we can see the equivalence with order books where the order is based on the order execution time: since there cannot be a shared clock due to the asynchronicity of the system, A bases its ”order book” on the puzzle resolution time.

4.2 The Token Exchange

We call Bob the winner of the previous round and we denote him as B. From now on, Alice will only interact with Bob and will discard all other potential exchangers.

The goal of this second round is to obtain a token exchange protocol that is atomic. In particular, if A has 1 `coin1`, and B has 1 `coin2` [3] then there can be only two succesful ending of the protocol: either$A$ has 1 `coin2` B has 1 `coin1` or A has 1 `coin1` B has 1 `coin2`.

Given the uncertainty of the real execution time to find a solution for a time-lock puzzle (the uncertainty is in the order of tens of seconds) it is not possible to carry out a simple exchange of keys between the parties. A few seconds are enough to issue two transactions, so the participant who first solves the assigned time-lock puzzle is able to take both the token associated with the solution and his token. Therefore the protocol would not be provably atomic. In the following we describe a method that allows to overcome this problem.

[3] The real exchange rates between the two tokens and how they are decided are beyond the scope of this paper.

Recall that given two secret keys sk_1 and sk_2, an elliptic curve generator g and the relative public key pk_1 and pk_2, then the key sum is homomorphic:

$$(sk_1 + sk_2)g = pk_1 + pk_2 = (sk_1 g) + (sk_2 g) \tag{1}$$

The Swap The actual exchange of tokens `coin1` and `coin2` is explained in Figure 4.3, with notation in Table 4.2. A and B create the key pairs (sk_2^A, pk_2^A and sk_1^B, pk_1^B respectively) that represent the first of the two shares to redeem the exchanged funds. After this step, A and B exchange the public keys pk_2^A and pk_1^B. If this first part is successful, both A and B create ephemeral keys (sk_1^A, pk_1^A and sk_2^B, pk_2^B respectively) that represent the second of the two shares to redeem the exchanged funds. At this point A and B can create new public keys (called PK_1^B and PK_2^A respectively) to which they can send the funds. Because of the way the keys and consequently the addresses are built, neither of the two participants can redeem the funds in either blockchains at this point of the exchange. For example, A needs to know sk_2^B in order to redeem coins in the address for PK_2^A. For this reason in the second part of the exchange A and B exchange the time-

BC_1,BC_2	Blockchain 1 and 2 with tokens `coin1` and `coin2`
G_1,G_2	the base point for the elliptic curve of BC_1 and BC_2
l_1,l_2	the base point order for the elliptic curve of BC_1 and BC_2
PK_1^A	public key for the address on blockchain BC_1 where A has the coins
PK_1^B	public key for the address on blockchain BC_2 where B has the coins
PK_2^A	public key for the address on blockchain BC_1 where A has the coins
PK_2^B	public key for the address on blockchain BC_2 where B has the coins
$sk_{1,2}^A, pk_{1,2}^A$	shares created by A for blockchian $BC_{1,2}$
$sk_{1,2}^B, pk_{1,2}^B$	shares created by B for blockchian $BC_{1,2}$

Table 1. Notation used in the explanation of the token exchange protocol

lock puzzles TLP_A and TLP_B. Once the time-lock puzzles are opened/solved, A gets sk_2^B and B gets sk_1^A. Using λ as time unit, we see that the second time-lock puzzle is sent later with a smaller opening time (1/4 of the time unit). This is to make the two participants A and B open the puzzle at about the same time.

4.3 Analysis of the BTLE protocol

As in the HTLC case, BTLE is also atomic in the sense that either both participants get tokens from the other blockchain, or both participants can retrieve their tokens. This is possible if we assume that all participants are rational and thus incentivized to respect different timeouts, as in the case of HTLC. The only step where there is a possibility of stealing the secret and breaking the atomicity, is the one where participant A has the time-lock puzzle of participant B, but A has not yet sent his time-lock puzzle. In this case A could start working on the received time-lock puzzle and discover the secret without sending his own time-lock puzzle. This case, however, is covered by the protocol: B waits a limited amount of time (a quarter of the expected time to solve the time-lock puzzle) and in case it does not receive A's puzzle, he would proceed to recover his tokens assuming A is dishonest. The only requirement is that the time lambda be longer than the time for creating new blocks in the blockchains between which the exchange takes place: that to prevent participant A from solving the received

Alice (`coin1`$\to$`coin2`)		Bob (`coin2`$\to$`coin1`)
$sk_2^A \xleftarrow{\$} [1, l_2 - 1], pk_2^A = sk_2^A G_2$		$sk_1^B \xleftarrow{\$} [1, l_1 - 1], pk_1^B = sk_2^B G_1$
	$\xrightarrow{pk_2^A}$	
	$\xleftarrow{pk_1^B}$	
$sk_1^A \xleftarrow{\$} [1, l_1 - 1], pk_1^A = sk_1^A G_1$		$sk_2^B \xleftarrow{\$} [1, l_2 - 1], pk_2^B = sk_1^B G_2$
$PK_1^B = pk_1^A + pk_1^B$		$PK_2^A = pk_2^A + pk_2^B$
$hash_{A\to B} \leftarrow$SendTx$(PK_1^A \to PK_1^B)$		$hash_{B\to A} \leftarrow$SendTx$(PK_2^B \to PK_2^A)$
$T_1 \leftarrow TLP_A(sk_1^A, \frac{3}{4}\lambda)$		$T_2 \leftarrow TLP_B(sk_2^B, \lambda)$
	$\xleftarrow{(T_2, hash_{B\to A})}$	
	$\xrightarrow{(T_1, hash_{A\to B})}$	
open TLP_B and redeem `coin2`		open TLP_A and redeem `coin1`

Fig. 1. Protocol execution between Alice and Bob for a successful swap

time-lock puzzle and sending the solution to take the money before the block is created, preventing B from acting safely, i.e. sending a transaction to retrieve its token. In this case both the transaction of A and the one of B would appear in the mempool of the miners/validators and it is not possible to know in advance which of the two transactions will end up in the block (invalidating the atomicity of the system). By choosing a suitable lambda the problem does not arise and it is not possible for A to steal the tokens.

Another advantage of BTLE over traditional cross-chain swaps is that it does not require the use of hash functions. This is because the BTLE uses techniques at a lower level than other atomic swap methods. In fact, we use the fact that the sum between points in elliptic curves is homomorphic, so BTLE is not affected by the internal mechanics of a blockchain protocol. This means that our protocol can also be used on blockchains that do not use the same hash function.

Using only elliptic curve theory, BTLE can also be used on blockchains that do not have a scripting language, such as Monero and all blockchains that are derived from CryptoNote. This additional advantage gives the possibility to implement BTLE in all those cases for which to date there are no exchange methods.

Finally, since there is little communication between participants, it is not necessary for all parties to be constantly online, unlike other methods. In fact, they can stay offline for the duration of solving a time-lock puzzle without this creating security problems within the protocol.

5 Conclusions and Future works

We presented in this paper the preliminary results of BTLE, a P2P and broadcast exchange protocol that creates an alternative to order-books and AMMs in a decentralized context. Unlike the other methods, it does not require the parties to be constantly online to finalize the exchange, thanks to the use of time-lock puzzles. We also proposed a new time-lock puzzle that is an alternative to the classical time-lock puzzle of Rivest et. al.

At this moment, we are better investigating the two types of proposed time-lock puzzles with the goal of understanding which method is more suitable for short duration puzzles (in the order of seconds). We are also working on the implementation of the BTLE protocol to demonstrate its applicability on different blockchain technologies, like Bitcoin, Monero, Ethereum.

References

[ACDP20] Isra Mohamed Ali, Maurantonio Caprolu, and Roberto Di Pietro. Foundations, Properties, and Security Applications of Puzzles: A Survey. *arXiv:1904.10164 [cs]*, April 2020.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. In *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 3-6, 1998, Proceedings*, pages 86–99. IEEE Computer Society, 1998.

[BBBF18a] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, volume 10991, pages 757–788. Springer International Publishing, Cham, 2018.

[BBBF18b] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.

[BCD$^+$14] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling Blockchain Innovations with Pegged Sidechains. *Whitepaper*, page 25, 2014.

[BG96] Mihir Bellare and Shafi Goldwasser. Encapsulated Key Escrow. Technical report, Massachusetts Institute of Technology, 1996.

[BM16] Emanuele Bellini and Nadir Murru. An efficient and secure RSA-like cryptosystem exploiting Rédei rational functions over conics. *Finite Fields and Their Applications*, 39:179–194, May 2016.

[BMRS18] Michael Borkowski, Daniel McDonald, Christoph Ritzer, and Stefan Schulte. Towards atomic cross-chain token transfers: State of the art and open questions within tast. *Distributed Systems Group TU Wien (Technische Universit at Wien), Report*, 2018.

[BN00] Dan Boneh and Moni Naor. Timed Commitments. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Mihir Bellare, editors, *Advances in Cryptology — CRYPTO 2000*, volume 1880, pages 236–254. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[But16] Vitalik Buterin. Chain interoperability. *R3 Research Paper*, 2016.

[Her18] Maurice Herlihy. Atomic Cross-Chain Swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254, Egham United Kingdom, July 2018. ACM.

[JMR20] Samuel Jaques, Hart Montgomery, and Arnab Roy. Time-release Cryptography from Minimal Circuit Assumptions. Technical Report 755, 2020.

[KB] Jae Kwon and Ethan Buchman. Cosmos Whitepaper. https://cosmos.network/cosmos-whitepaper.pdf.

[m5221] m52go. Bisq Whitepaper, June 2021. original-date: 2017-09-06T18:19:27Z.

[MD19] M. H. Miraz and D. C. Donald. Atomic Cross-chain Swaps: Development, Trajectory and Potential of Non-monetary Digital Token Swap Facilities. *Annals of Emerging Technologies in Computing*, 3(1):42–50, January 2019.

[MMS$^+$19] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic Time-Lock Puzzles and Applications. In *Advances in Cryptology – CRYPTO 2019*, volume 11692, pages 620–649. Springer International Publishing, Cham, 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Whitepaper*, page 9, 2008.

[NFSD20] Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker. Toward Active and Passive Confidentiality Attacks On Cryptocurrency Off-Chain Networks. *arXiv:2003.00003 [cs]*, February 2020.

[Nol] T. Nolan. Alt chains and atomic transfers. https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949.

[Pie18] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[Poe15] Andrew Poelstra. On Stake and Consensus. Technical report, March 2015.

[Poo17] Joseph Poon. Plasma : Scalable autonomous smart contracts. 2017.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release Crypto. Technical report, 1996.

[Rus19] Russel Rusty. lightningnetwork/lightning-rfc, September 2019. original-date: 2016-11-14T19:21:45Z.

[Wes20] Benjamin Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020.

[Woo16] Dr Gavin Wood. POLKADOT: VISION FOR A HETEROGENEOUS MULTI-CHAIN FRAMEWORK. *Whitepaper*, 2016.

[WPG$^+$21] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. Sok: Decentralized finance (defi). *CoRR*, abs/2101.08778, 2021.

[XVP$^+$21] Jiahua Xu, Nazariy Vavryk, Krzysztof Paruch, Simon Cousaert, et al. Sok: Automated market maker (amm) based decentralized exchanges (dexs). Technical report, 2021.

[ZAZ$^+$18] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J Knottenbelt. SoK: Communication Across Distributed Ledgers. *ePrint IACR*, page 17, 2018.