R-SWAP: Relay based atomic cross-chain swap protocol

1st Léonard Lys
LIP6, Sorbonne Université
PALO IT
Paris, France
llys@palo-it.com

2nd Arthur Micoulet
PALO IT
Paris, France
amicoulet@palo-it.com

3rd Maria Potop-Butucaru
LIP6, Sorbonne Université
Paris, France
maria.potop-butucaru@lip6.fr

***Abstract*—In this paper, we consider the problem of cross-chain transactions where parties that do not trust each other safely exchange digital assets across blockchains. Open blockchains models are decentralized ledgers that keep records of transactions. They are comparable with distributed account books. While they have proven their potential as a store of value, exchanging assets across several blockchains remains a challenge. Our paper proposes a new protocol, R-SWAP, for cross-chain swaps that outperforms existing solutions. Our protocol is built on top of two abstractions: relays and adapters that we formalize for the first time in this paper. Furthermore, we prove the correctness of R-SWAP and analytically evaluate its performances, in terms of cost and latency. Moreover, we evaluate the performances of R-SWAP in two case studies showing the generality of our approach: atomic swaps between Ethereum and Bitcoin (two popular permissionless blockchains) and atomic swaps between Ethereum and Tendermint (one permissionless and one permissioned blockchain).**

***Index Terms*—Blockchain, atomic swap, cross chain transactions, relays**

I. Introduction

With the raise of digital assets hosted on blockchain systems came the need for exchanging them across chains. Today, most of the cross-chain exchanges are achieved with the help of a trusted third party, most often an exchange platform. Centralized exchange platforms (CEX's) don't support peer-to-peer cross-chain transactions between users. Instead, users deposit their funds inside the platform's digital wallets and the transactions are handled by the platform's system. Most of the transactions are not even published to the chain. Platforms keep a separate record of their clients assets; the only transactions that are written to the blockchain are deposits and withdraws. This is not a satisfactory solution, because as stated in Bitcoin's white paper [1] all benefits of blockchain technologies are void if a centralized financial institution is required. Indeed, trusting a centralized third party comes with all the flaws of centralization: attractive target for attackers [2], governance issues [3], platform commissions, etc. Thus there is a need for solutions that allow users to perform trust-less, cross-chain, peer-to-peer, atomic transactions.

Atomic cross-chain swaps are a solution to this problem. They are distributed protocols where several parties exchange assets across chains. An Atomic cross-chain swap protocol must ensure safety and liveness, i.e., no participant complying with the protocol will lose money and if both participants abide by the protocol, they eventually get their payoffs or get refunded.

*a) **Hash time locking**:* A proposed strategy for atomic cross-chain swaps relies on hash time locking [4]. This technique consists of locking some digital assets inside a smart contract or script and set the release condition to both a time condition and the revelation of a cryptographic hash pre-image. Herlihy's protocol [5] for instance, makes use of this technique. The protocol is as follows: Alice has X assets of chain BC_a, Bob Y assets of chain BC_b and they are willing to exchange one for the other. They agreed on the exchange rate trough off-chain communication channels (1). Alice creates a random secret s and computes its hash $h = H(s)$ (2). She publishes a contract SC_1 on blockchain BC_a where she locks her assets and set the release condition to the revelation of the pre-image of h or the expiration of a time lock Δ_1 (3). Bob learns h from SC_1 and sets up a similar contract on BC_b with a time lock $\Delta_2 < \Delta_1$ (4). Alice redeems SC_2 by revealing s and by doing so triggers the transfer of Y from SC_2 to her address (5). Bob learns about s from this transaction and can now proceed to redeem X from SC_1 (6). If for some reason, a participant stops in the process, they can safely wait for the time lock to expire and proceed to refund.

Current proposed protocols that use the hash time locking technique such as [5] are subject to several limitations. They suffer from a potential safety violation, they may be economically unfair and have bad ergonomics [6], [7]. Indeed, after stage (5), if for some reason Bob is unable to broadcast his redeem transaction message (client's crash, DDOS attack on the client or network, packet loss, etc.), Alice can wait for the time lock to expire, send a refund transaction message and retrieve both assets, causing a safety violation. Another weakness is that after stage (3), Bob can choose not to publish a contract SC_2, causing Alice's assets to be locked until the expiration of Δ_1 [8]. This is not a proper liveness violation, but in an highly competitive market such as crypto-assets trading, having an adversary's funds locked up for a certain amount of time is a great advantage. Furthermore, the cost of such an attack is null for Bob given that at no point he had to prove ownership of assets. Finally, this protocol has poor ergonomics; it requires both participants to stay online during

the swap, to run two blockchain clients each and to perform auditing tasks and transactions on both chains.

b) ***Relays****:* An other strategy to achieve atomic cross-chain swaps relies on *relays* [4], [9], [10]. Relays are abstractions (in general a smart contract or a script) hosted on some chain BC_a that has light client like verification capabilities over chain BC_b. For each new block appended to chain BC_a, the block header is passed on to the relay on chain BC_b. The relay itself implements the standard verification procedure of chain BC_a's consensus algorithm and can therefore verify the validity of the block. Once the proof of work has been verified, in the case of a Proof of Work (or PoW) blockchain, or the two-thirds of validators signatures, in the case of a Byzantine Fault Tolerant (or BFT) blockchain, it is possible to verify any transaction of chain BC_a from chain BC_b. With light client like verification capabilities of chain BC_a from chain BC_b, we can imagine the following scenario. Bob has X assets of chain BC_b. He is willing to exchange them for Y assets of chain BC_a. Bob sets up a smart contract SC_1 and locks his assets in it (1). This smart contract SC_1 is set to release the assets to anyone providing the proof that they made a payment of Y assets of chain BC_a to Bob's address. Alice, who is interested in this trade, transfers Y assets to Bob's address (2). She retrieves the transaction hash tx and provide it to SC_1 (3). SC_1 calls the relay and asks for verification of transaction tx (4). The relay verifies that the transfer has taken place and if so, returns ok to SC_1 (5). On receiving the answer from the relay, SC_1 transfers the X assets of BC_b to Alice's address.

Though this approach seems like a reasonable solution to achieve atomic cross-chain swaps, it has several flaws that can lead to safety violation, liveness violation and commercial unfairness. The potential safety violation comes from a race condition type of attack. Indeed at stage (2), it is possible that an other person, for instance Carol, is interested in the swap and sends some assets to Bob's address. Alice and Carol will only see the other one's transfer when the next block is mined. Therefore there is no way for them to coordinate their transfers. Bob will end up with twice the assets he wanted and he has no incentive to refund the loosing party. The potential liveness violation comes from the fact that after stage (1) Bob has absolutely no guarantee that someone will take the swap. Therefore his assets could be locked up forever leading to a liveness violation. Finally the protocol is commercially unfair. Indeed, after stage (1) the rate of the swap is fixed and the assets locked up. With the high volatility of cryptocurrency markets, Alice can safely wait for the price to evolve in a direction that is profitable to her and then decide to perform the swap or not. This is commercially unfair to Bob.

c) ***Cryptocurrency-backed assets****:* The most recent proposed strategy for cross-chain swap is cryptocurrency-backed assets (or CBAs) such as [7]. Cryptocurrency-backed assets are tokens issued by a smart contract on chain I that are backed at a one to one ratio by assets locked on chain B. Thus each token $i(b)$ hosted on chain I represents a unit of asset of chain B. Each token $i(b)$ is backed by an asset on chain B held in custody. In order to issue a token $i(b)$, one must provide a proof that a unit of asset of chain B has been locked in a *vault* (the custody) . The proof, the locking transaction, is then verified thanks to a blockchain relay. Once the proof is verified, the token $i(b)$ is issued. As they now exist in the same self-contained blockchain system, $i(b)$ tokens can be atomically swapped with assets of chain I (like ERC-20 tokens). In order to recover the assets locked on chain B, one must *burn* an equivalent amount of token $i(b)$. This proof-of-burn is then submitted to the *vault*, that will redeem the assets. To ensure that the vault behaves honestly and redeems the assets, it must lock assets of chain I in the smart contract as collateral. A single *vault* will only be able to issue as much $i(b)$ tokens as it has locked assets of I in collateral, times the exchange rate. If the exchange rate ϵ between assets of I and B becomes critical (i.e., if the collateral value is too low to reimburse the token owners), the collateral is liquidated and sent to the token owners as payback.

Although protocols such as [7] show good performance, reasonable costs and allow asset portability across chains, they do not solve the problem of atomic cross chain swaps. Such protocols can be assimilated with sidechain pegging like in [11]. The ownership of the assets of chain B is not transferred. Instead, a token of equal fiat value is issued on chain I. Moreover strong assumptions are made on the price oracle. It is assumed in the model that the exchange rate ϵ between assets of B and I given by the oracle is correct. Under weaker assumptions (i.e, if the exchange rate returned by the oracle is not correct), a safety violation could occur. The liquidation of the collateral is triggered by the exchange rate ϵ given by the price oracle. If an attacker is able to achieve an "Oracle poisoning attack", he could trigger liquidation of the collateral and lead to a safety violation. Such attacks have already happened [12], [13]. Our proposal does not suffer from this kind of risks because price oracles are not involved.

d) ***Our contribution****:* Our contributions are as follows: we propose a new protocol for atomic cross-chain swap, that does not suffer from the limitations of the previous protocols in the literature (e.g. safety violation as in the case of [5] or unfairness issues [6], [7]). Moreover, we propose a formalisation and correctness proof of our protocol. Furthermore, we evaluate analytically its cost and latency. Furthermore, we evaluate the performances of our protocol considering two case studies: atomic swaps between Bitcoin and Ethereum and atomic swaps between Tendermint and Ethereum. Interestingly, our protocol builds on top of two abstractions that we formalized for the first time: blockchain *relays* and *adapters*. These abstractions can be of independent interest.

e) ***Paper roadmap****:* Section II introduces the system model. In Section III we propose a formalisation for the atomic cross-chain swap, blockchain adapter and relay abstractions. In Section IV and Section V we propose a detailed description of our R-SWAP protocol altogether with its correctness proof. Furthermore, in Section VI we analytically evaluate the performances of our protocol. Finally Section VII presents some conclusions and discussions.

II. Model and definitions

A. System model

In this paper, we consider open blockchain systems such as Bitcoin, Ethereum or Cosmos. Each blockchain system is composed of an arbitrary finite set of processes Π namely $\Pi = \{p_1, p_2, ...\}$. The size of the set Π is not known. Moreover processes can crash, leave or join the network at any moment.

*a) **Communication**:* Processes exchange messages trough peer-to-peer, bi-directional communication channels. Each process p_i is connected to a subset of peers $\Pi_p \subset \Pi$. The system uses a gossip protocol for message broadcasting. When a process p_i wants to broadcast a message msg, he sends it to each process of subset Π_p. Each correct process forwards the message to its own subset of neighbors $\Pi_{p'}$, until eventually every correct processes of Π delivered the message. To simplify, we introduce a primitive BROADCAST(TAG, msg), where TAG is the type of message and msg the actual message. Processes invoke the DELIVERY primitive to receive messages. We assume that when a correct process invokes the BROADCAST(TAG, msg) primitive, every correct process eventually delivers it. However, as messages can be delayed for an arbitrary long time, we assume asynchronous communication.

*b) **Authentication**:* Each processes in the system possesses a pair of public/private keys. Messages are authenticated by digital signature. We consider that it is impossible to forge the signature.

*c) **Failure model**:* The size of the set Π is not known. Moreover processes can crash, leave or join the network at any moment. We say that a process exhibits Byzantine behaviour when it behaves arbitrarily [14]; for example not relaying messages, delaying them or broadcasting inconsistent messages. A Byzantine process is said to be faulty, otherwise it is a correct process. We assume that there can be as much as $f < n/3$ faulty processes.

B. Distributed Ledger Model

A distributed ledger (a.k.a. blockchain) BC is an append-only list of blocks chained together. Each block b_i somehow contains a list of transactions as well as a variable pointing to the previous block b_{i-1}, hence the name blockchain. The blockchain supports two types of operations READ(BC) and APPEND(BC, b_i).

We consider two types of processes, miners and clients. Miners manage a local copy of the blockchain, they participate in the gossip protocol and they produce new blocks. Clients can maintain a copy of the blockchain or synchronise it with a miner. As the system is distributed, the "global" state is composed of the sum of all "local" states of each miner [15]. By listening to incoming transaction messages, miners build a pool of transactions that have not been included in a block yet.

If at time τ a miner process p_i has been selected to produce a new block b_i, he broadcasts it by invoking BROADCAST$(BLOCK, b_i)$. The consensus mechanism is such that, if p_i is correct, at time $\tau' > \tau$ eventually every READ(BC) operation on a correct miner will return BC where $b_i \subset BC$.

*a) **Ledger conflict**:* A ledger conflict (or fork) is an event during which two concurrent blocks have been found at the same height h. Due to the system being asynchronous, if two miners find respectively blocks b and b' nearly at the same time, then both b and b' will propagate, resulting in a network partition. A partition P that appended b to their copy of the blockchain and another P' that appended b'. A fork is resolved when subsequent block(s) are found. The network will choose the longest chain (in terms of height). Any block that is not in this longest chain will be abandoned. For example if partition P that has appended b finds a new block at height $h+1$ before partition b' can do so, then members of P' will abandon b' and replace it with b (and any subsequent blocks found by P).

*b) **k-safety**:* The parameter k is the depth at which the probability of a fork or ledger conflict is sufficiently low to consider that the risk of this block being abandoned is null. Indeed, because the system is asynchronous, it is difficult for a node to determine if the network is currently partitioned or not (i.e., if a fork or ledger conflict has happened). Thankfully, forks tend to resolve as new blocks are mined. So the deeper a block is, the safer it is to assume that it is in the others node's copy of the chain. After k new blocks, it is safe to consider that no forks occurred prior to the said block. The parameter k depends on the blockchain you are using and the level of safety your system requires. For example most Bitcoin clients consider $k = 6$ to be safe, but some old versions would require as much as k=120, because at that time the risk of fraudulent fork (e.g 51% attack) was higher.

*c) **k-Valid block**:* A block that has been appended to the chain and that is now at a depth equal or greater than k. Each new block appended to the chain after the block of interest is called a confirmation.

*d) **Confirmed transaction**:* A transaction that has been included in a block. A confirmed transaction is not necessarily valid yet.

*e) **k-Valid transaction.**:* A transaction is said to be k-valid when it is included in a k-valid block, i.e it is included in a block that has received k confirmations.

*f) **Simple Payment Verification (SPV)**:* Simple payment verification is a process that allows a client to verify the validity of a transaction without having to maintain a full copy of the blockchain. Instead, the client only needs a list of block headers. Implementation of this process may vary depending on the blockchain.

*g) **Block time**:* The block time is the average block creation time. It is calculated by dividing a large period of time by the number of blocks produced during this period. This parameter is specific to each blockchain. Because block creation is probabilistic in PoW consensus blockchain, block creation time can vary from one block to the other. Average block time is obtained by adjusting the difficulty [15]. In Bitcoin for example, the target block time is 10 minutes. BFT

consensus blockchains are deterministic, hence they have a fixed block time [16].

*h) **Transaction validation time**:* Let process p broadcast a transaction tx via the primitive BROADCAST($< transaction >, tx$) at time $t_{broadcast}$. Let t_{valid} be the time when the transaction tx appears to be k-valid to p. Then we define the transaction validation time as:

$$\Delta_{validation} = t_{valid} - t_{broadcast} \quad (1)$$

C. Asset model

*a) **Blockchain asset**:* As of today, blockchain asset refers to anything on the blockchain that serves as store of value or medium of exchange. It can be coins such as Bitcoins and Ethers, or it can be tokens such as ERC-20 tokens. Nevertheless blockchain assets share the following properties; A blockchain asset belongs to a public/private key pair. Asset ownership can be transferred trough transactions. Asset transfers are recorded to the chain. To transfer the ownership of an asset, one must sign the transaction with the private key.

*b) **Asset locking**:* An asset on a blockchain is said to be locked when it is not possible to transact it without meeting some conditions. Accessing the said conditions would unlock the asset and allow the user to transact it. Assets are locked thanks to scripts or smart contracts.

*c) **Time locking**:* An asset is time locked when the condition required to transact the asset is a time condition.

*d) **Hash locking**:* An asset is hash locked when the condition required to transact the asset is the revelation of the preimage of a cryptographic hash (hash key).

*e) **Acceptable Payoff**:* When trading an asset for another, the payoff is said to be acceptable when it is equal to the amount of asset sent times the exchange rate agreed upon, minus the transaction fees.

III. Time locked atomic cross-chain swap

In this section we define the time locked atomic cross-chain swap problem.

A. Problem specification

A time locked atomic cross-chain swap protocol uses time locking in order to ensure termination of the protocol. At the expiration of the time lock, both participants have received an acceptable payoff.

Definition 1 (time locked atomic cross-chain swap) A time locked atomic cross-chain swap protocol should satisfy the following properties:

- **Safety.** No participant abiding by the protocol can lose more money than the transaction fees.
- **Time-bounded termination.** No asset can be locked for more than a period of time γ and if an asset is locked, γ is known prior to locking.
- **Liveness.** Upon lock time expiration, if all participants abided by the protocol and no failures occurred then they received their payoffs.

B. Abstractions for implementing R-SWAP

*a) **Blockchain adapters**:* A Blockchain adapter is a piece of software that allows to send transaction and query several blockchain systems. In addition to its multiple-blockchain client capabilities, it can perform off-chain computation and storage. It is able to store private information such as private keys and credentials [17]. A blockchain adapter can be hosted by several instances. The systems goals of a blockchain adapter are:

- Client capabilities: A blockchain adapter should be able to read the state of at least two blockchains, as well as broadcasting transactions.
- Off-chain computation: A blockchain adapter should be able to perform off-chain computations, such as generating a random value and computing its hash.
- Wallet safety: A blockchain adapter should be able to store private credentials and key, without the person running it being able to extract those private data.
- Decentralization: A blockchain adapter should be replicated over several nodes

*b) **Blockchain Relay**:* A blockchain relay $R_{a \leftarrow b}$ is an abstraction (smart contract or script) on chain BC_a that can receive verification requests of transactions on chain BC_b. It receives block headers of chain BC_b and performs the standard verification for blocks of BC_b (1). It stores block headers of chain BC_b (2). It can perform Simplified Payment Verification over transactions on chain BC_b and either returns true if the transaction is valid or false if it is not (3).

Definition 2 (Blockchain Relay) A blockchain relay must satisfy the following properties:

- **k-Validity.** A blockchain relay returns true if the submitted transaction is a k-valid transaction of blockchain BC_b, otherwise it returns false.
- **Eventual Persistent Storage.** Every valid block header of chain BC_b eventually ends up being included in a block appended to BC_a.

A blockchain relay hosted on BC_a that performs verification over blockchain BC_b is noted $R_{a \leftarrow b}$.

Algorithm 1 Blockchain Relay smart contract class

```
1: bytes [] blocks                          ▷ Table to store block headers
2: procedure STOREBLOCKHEADER(blockHeader)
3:     requires(VALIDATEPOW(blockHeader))
4:     THIS.BLOCKS.APPEND(blockHeader)      ▷ If the block is valid, store block
5: end procedure
6: procedure VERIFYTX(tx, k)
7:     return VALIDATESPV(tx.txid, tx.root, tx.proof, tx.index, k)
8: end procedure
```

Algorithm 1 is a possible interface for a blockchain relay. It presents the main functions of such a software. The function VALIDATEPOW (line 3) implements the Proof of Work verification algorithm of the relayed blockchain. It returns true if the provided block header is valid and false otherwise. The function VALIDATESPV (line 7) implements the simple payment verification of provided transaction. It returns true if it is a valid transaction and false otherwise. The parameter k in function VERIFYTX(tx, k) is the safety factor. It specifies at what height the block containing the transaction tx must be in the relay's data store to consider the transaction as valid. Complete implementation of VALIDATEPOW and VALIDATESPV can be found in [18].

*c) **Blockchain Relay latency**:* We define blockchain relay latency Δ_{relay} as the time difference between the moment a new block b_i is mined on chain BC_b and the moment a verification request for $tx_i \subset b_i$ to the relay $R_{a \leftarrow b}$ with safety parameter k will return true. Let t_{mined} be the time at which a relayer process p of blockchain BC_b delivers a block b_i via the primitive DELIVER. Let t_{valid} be the time when VERIFYTX(tx_i, k) will return true. Then the blockchain relay latency is given by:

$$\Delta_{relay} = t_{valid} - t_{mined} \tag{2}$$

IV. R-SWAP PROTOCOL

This section presents R-SWAP, a protocol that achieves time locked atomic cross-chain swaps.

A. Protocol Overview

The R-SWAP protocol relies on three software bricks; hash time locked contracts (Algorithm provided in appendix E), blockchain relays (Algorithm 1) and blockchain adapters. Each participant has an instance of an adapter. Each adapter is able to send transactions on both blockchain BC_a and BC_b. The unfolding of R-SWAP resembles a hash time locked contract (HTLC) atomic swap protocol but with two major improvements. With HTLC swaps, the users were responsible for contract and blockchain auditing. In R-SWAP this is done thanks to the relay's cross-chain verification capabilities. HTLC based swap required the user to perform complex tasks such as generating hash locks, time locks and sending several transactions on both chains. R-SWAP uses adapters that automate all those tasks. With R-SWAP the number of user actions is limited to one.

In a first phase, participants will commit to the swap by locking their assets with smart contracts. In a second phase, participants will receive their payoffs or refunds.

B. Phase 1: Commitment phase

When Alice wants to execute an atomic cross-chain swap with the R-SWAP protocol, she will request her adapter to lock up funds inside a hash time locked contract SC_1. This contract SC_1 exists in one of five states: *Invalid*, *PreCommitted*, *Committed*, *Redeemed* and *Refunded*. State changes and the corresponding asset transfers are triggered by the following functions: PRECOMMIT, COMMIT, REDEEM, REFUND. State changes are unidirectional, thus SC_1 can be seen as a directed acyclic graph where each state is a node and each vertex a state change function (see Figure 3 Appendix A).

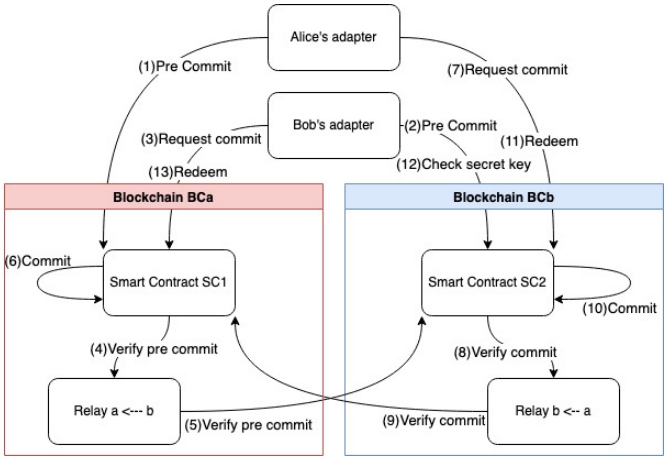

Fig. 1. High level overview of the R-SWAP components and interactions

Alice only specifies to the adapter the exchange rate and the amount. The adapter will be responsible for generating a random secret s and to compute its hash $h = H(s)$. Then the adapter will lock up the funds inside SC_1 hosted on chain BC_a by calling the PRECOMMIT function.

At this point, the hash lock has been specified, but not the time lock. Hence the assets are not locked, because Alice can ask for a refund at any moment. The PRECOMMIT state serves as a proof of ownership.

It is possible to implement an interface that would list all contracts in *PreCommitted* state and hence build a decentralized trading platform. This platform would list open swaps and their parameters hashlock, amount and exchange rate. Bob, a participant interested in the swap, would just have to retrieve the parameters from the decentralized trading plateform and request his adaptor to set up a mirror contract SC_2 on blockchain BC_b.

Once the funds are hash locked in SC_2 trough PRECOMMIT function, Bob will request that the contract SC_1 sets him as the receiver and define a time lock. To do so, Bob's adapter will call the COMMIT function of SC_1, with as parameter, the transaction hash of SC_2's PRECOMMIT. Indeed before setting Bob as receiving party, SC_1 needs the proof that some funds have been locked up in parallel on BC_b. SC_1 will call the *verifyTx* function of the relay $R_{a \leftarrow b}$, with as a parameter, the PRECOMMIT transaction of SC_2 provided by Bob's adaptor. If the transaction is valid, $R_{a \leftarrow b}$ will return the parameters of the transaction.

Here the relay $R_{a \leftarrow b}$ serves as a bridge between BC_a and BC_b. It allows contract auditing to be made directly on chain by SC_1 and SC_2. The authenticity of the data is ensured and hence the safety of the protocol improved. Once SC_1 has verified SC_2's parameters, it can safely set Bob as the receiver, Δ_1 as time lock and change state to *Committed*. SC_1 sends back the transaction details to Bob's adapter, that redirects it

to SC_2. SC_2 calls $R_{b \leftarrow a}$ to verify SC_1's commitment, and if so commits with Alice as a receiver and Δ_2 as time lock.

Here is a sum up of the commitment phase.

1) Alice calls her adapter's PRECOMMIT function with target network, value and exchange rate
2) Alice's adapter generates secret s and computes $h = H(s)$
3) Alice's adapter calls SC_1 PRECOMMIT function
4) Alice's adapter runs a daemon
5) Bob calls the PRECOMMIT function of his adapter with Alice's mirror parameters
6) Bob's adapter calls SC_2 PRECOMMIT function
7) Bob's adapter calls SC_1 COMMIT function
8) SC_1 calls the relay $R_{a \leftarrow b}$ to verify that SC_2 is in state *PreCommitted*
9) On receiving the proof that SC_2 is in state *PreCommitted* SC_1 changes its state to *Committed* sets Bob as receiver and Δ_1 as time lock
10) On receiving proof that SC_1 is *Committed*, Bob's adapter calls SC_2 COMMIT function.
11) SC_2 calls the relay $R_{b \leftarrow a}$ to verify that SC_1 is in *Committed* state
12) On receiving the proof that SC_1 is on *Committed* state, SC_2 changes its state to *Committed* sets Alice as receiver and Δ_2 as time lock
13) Bob's adapter runs a daemon

C. Phase 2: Contracts redeem/refund

a) ***Daemon****:* At stage 4 and 13 each adapters have run a daemon. Those daemons are awaiting for a state change of SC_2, before they move on to the redeem/refund phase. For each new block of BC_b Alice's daemon will check if Bob's contract SC_2 has changed from state *PreCommitted* to *Committed*. This would mean that Bob has committed to the swap and thus that she can move on to the redeem phase. For each new block of blockchain BC_b Bob's daemon will check if SC_2 has changed from state *Committed* to *Redeemed*. This would mean that Alice has redeemed, and by doing so, revealed the secret key. Bob can now move on to the redeem phase.

Those daemons also handles the cases where some party would stop abiding by the protocol. They have stored the time lock values and a REFUND transaction call is scheduled at the expiration of the time lock.

Here is a sum up of the second phase, if participants abide by the protocol:

1) Alice's adapter calls SC_1 CHECKSTATE function
2) If state is *Committed* call SC_2 CHECKSTATE function
3) If state is *Committed* Alice's adapter calls REDEEM function of SC_2 with s, triggering the asset transfer to her address
4) Bob's adapter calls SC_2 CHECKSECRETKEY function
5) If key is revealed Bob's adapter calls SC_1's REDEEM function with the secret s he just learned, triggering the asset transfer to his address

If one of the participant does not abide by the protocol, the daemon will simply call REFUND at the expiration of their respective smart contract's time lock expiration.

V. R-SWAP PROTOCOL CORRECTNESS

A. Time lock value determination

Choosing the right value for the time locks is essential. A time lock value that is too short could lead to a safety violation. Conversely, a time lock value that is too long could be commercially unfair, as prices are very volatile.

Consider a swap executed via the relays swap protocol. The swap is executed between blockchains BC_a and BC_b. k_a resp. k_b is the k parameter of BC_a resp. BC_b (see k-safety in Section II-B0b). ζ_a is the target block time of BC_a and ζ_b of BC_b. There are two time locks values to be chosen; Δ_1 for SC_1 and Δ_2 for SC_2.

The first value of interest is the difference between Δ_1 and Δ_2. This difference has to be long enough for the participant to call the redeem function and for this transaction to be included in a valid block. If not long enough this could lead to a safety violation. Δ_1 and Δ_2 are strongly dependent on parameters specific to each blockchain. A naive estimation of the minimal value for the difference between time locks could be:

$$\Delta_1 - \Delta_2 = (\zeta_b k_b) \tag{3}$$

Assuming synchronous communication, this should be long enough for the redeem transaction to be included in a valid block. But as stated in the model, the block time is not necessarily fixed and blocks can be delayed for an arbitrarily long time, leading to potential safety violations.

Therefore for the rest of this paper, we will assume that there is a time λ, specific to each blockchain, within which a broadcast transaction will be included in a block and receive k confirmations with a probability ϵ. Thus $\Delta_1 - \Delta_2 \geq \lambda_b$ is the first constraint to satisfy the safety property, where λ_b is the upper bound on transaction validation for blockchain BC_b.

The second constraint does not concern safety, but commercial fairness. Bob will not call the COMMIT function of SC_2 unless he is sure that SC_1 is *Committed*. He needs the guarantee that Alice's commitment is a valid transaction. If it is not yet, in the fear of a safety violation, Bob should chose not to COMMIT and instead call REFUND on SC_2. Alice would end up having her assets locked for at least λ_b, which is commercially unfair for her. Thus, for the swap to be fair, time-locks should be chosen such that $\Delta_1 \geq \Delta_2 + \lambda_a$.

We obtain the following:

$$\Delta_1 \geq \Delta_2 + max\{\lambda_a, \lambda_b\} \tag{4}$$

Now that we found the minimal value for Δ_1, we want to find a minimal value for Δ_2. We want to minimize the duration of the swap for commercial fairness. If $\Delta_2 = 0$, Bob can call for a REFUND on SC_2 at any moment. Yet he cannot ask for REDEEM on SC_1 either he doesn't know s. Alice's adapter has been triggered by Bob's COMMIT and proceeds to call REDEEM on SC_2. This REDEEM transaction

contains s in plain text as a parameter. At this point this redeem transaction is not confirmed yet, but still waiting in the mempool. As transactions are public, it is possible for Bob to run a transaction sniffer that would extract s from Alice's unconfirmed REDEEM transaction. Then he would send two transaction: REFUND on SC_2 and REDEEM on SC_1 with the secret s he just sniffed, leading to a safety violation. He could even increase its chances of success by setting high transaction fees.

Therefore it is necessary that Δ_2 be long enough for Alice to call REDEEM on SC_2 and for this transaction to be included in a valid block. Thus we have:

$$\Delta_2 \geq \lambda_b \tag{5}$$

*a) **Relay latency**:* Cross chain transaction verification requires that, for each new block b_i produced on chain BC_b, a relayer submits b_i to the relay $R_{a\leftarrow b}$. Thus, between the time a block b_i is produced on chain BC_b and the time VERIFYTX(tx), $tx \subset b_i$ to the relay $R_{a\leftarrow b}$ will return true, there is a maximum delay given by [7].

$$\Delta_{relay} = \lambda_b + \Delta_{submit} + 2\lambda_a \tag{6}$$

λ_a and λ_b being the upper bound on transaction validation time of blockchain BC_a resp. BC_b. Δ_{submit} is the delay between the moment a block is produced and the moment a relayer submits the block to the relay.

In the R-SWAP protocol, each relay $R_{a\leftarrow b}$ and $R_{b\leftarrow a}$ is called once. SC_1 calls VERIFYTX(tx, k) at stage 8 to verify that SC_2 is in state *PreCommitted*. SC_2 calls the relay $R_{b\leftarrow a}$ at stage 11 to verify that SC_1 is in state *Committed*.

Let $tx1_{com}$ be the commitment transaction of SC_1. There is a latency Δ_{relay} during which the relay $R_{b\leftarrow a}$ will not consider $tx1_{com}$ as a valid transaction. Thus calling VERIFYTX$(tx1_{com}, k)$ on $R_{b\leftarrow a}$ during this period of time might return false. Thus the time lock value Δ_1 of contract SC_1, must be such that $\Delta_1 \geq \Delta_{relay}$. Considering relay's delay we must now ensure that Δ_1 is large enough for the commit transaction of SC_1 to be valid regarding the relay:

$$\Delta_1 \geq \Delta_2 + \Delta_{relay} \tag{7}$$

or

$$\Delta_1 \geq \Delta_2 + \lambda_a + \Delta_{submit} + 2\lambda_b \tag{8}$$

Finally we obtain the following system of inequalities for the time lock values:

$$\begin{cases} \Delta_1 \geq \Delta_2 + \lambda_a + \Delta_{submit} + 2\lambda_b \\ \Delta_2 \geq \lambda_b \end{cases} \tag{9}$$

B. Proof of correctness

In this section, we prove the correctness of the protocol, i.e., that it satisfies the properties of safety, time bounded termination and liveness of a time locked atomic cross chain swap, defined in Section III-A.

Lemma V.1. *Assuming Bob's adapter is correct and that there is an upper bound on transaction validation time λ, the R-SWAP protocol satisfies the γ time-bounded termination property of a time locked atomic cross-chain swap.*

Proof. Consider a time bounded atomic cross-chain swap executed via the R-SWAP protocol and assume there is a violation of the time bounded termination property. A violation of the time bounded termination property implies that the asset is locked, thus excluding all set of states prior to *PreCommitted/PreCommitted*, *PreCommitted/PreCommitted* included. Indeed, prior to be in state *Committed*, contracts are only hash-locked and thus can be refunded. Given the possible set of states in Figure 4 appendix B, subsequent possible states are *Committed/PreCommitted*, *PreCommitted/Refunded* and *Refunded/PreCommitted*. Those two last set of states don't imply a violation of the time bounded termination property since asset are refunded, or will be able to be refunded when the time lock elapse, so the only case to consider is the set *Committed/PreCommitted*.

In order to transition to this set, SC_1 must commit. It does so by calling the COMMIT internal function with as a parameter Δ_1. Δ_1's value is hard coded in the contract thus a value unknown or superior to γ is impossible. The result of the SC_1's commitment is a transaction tx.

Subsequently, three sets of state are possible:

- *Refunded/PreCommitted*: Since Δ_1 is known and finite, SC_1 will eventually be able to move to $Refunded$, not leading to a violation of the time bounded termination property.
- *Committed/Committed*: To proceed to state $Committed$, SC_2 calls $R_{b\leftarrow a}$ to verify tx. If the relay returns true then SC_2 extracts Δ_1 from tx. It then proceeds to call it's own COMMIT function with as a parameter $\Delta_2 = \lambda_b$. Yet a violation of the time bounded termination property would imply that the time lock value was unknown, leading to a contradiction.
- *Committed/Refunded*: Since its asset was not locked, SC_2 can be refunded. But since Δ_1 is known and finite, $SC1$ will eventually be able to move to $Refunded$, not leading to a violation of the time bounded termination property.

Out of those three sets of states, the only one of interest is *Committed/Committed*, as the other two involve SC_1 being able to call REFUND. Since Δ_1 is known and inferior to γ, SC_1 will be able to call REFUND at the expiration of Δ_1. Thus, from now on, the potential violation of the time bounded termination property only concerns SC_2. From the set of state *Committed/Committed* two set of states are possible:

- *Committed/Redeemed*: Since Bob has extracted h from Alice's COMMIT transaction tx, she can call the REDEEM function of SC_2 with the secret s. By doing so she triggers the asset transfer from SC_2 to her address. As she has revealed s to Bob, if Bob's adapter is correct it will automatically call the REDEEM function of SC_1, triggering the asset transfer from SC_1 to Bob's address. Since the transaction has a high probability of being included prior to Δ_2's expiration, the protocol satisfies the γ time bounded termination property.

- *Refunded/Committed*: If Alice has been refunded, it means that Δ_1 has expired. If Bob's adapter is correct, it would have called the REFUND function of SC_2 right after the publication of the block containing Alice's refund transaction. Since $\Delta_1 \geq \Delta_2 + \Delta_{relay}$, Bob's REFUND transaction will be included with a high probability, leading to a contradiction. Thus the protocol satisfies the γ time bounded termination property.

□

Lemma V.2. *Assuming Bob's adapter is correct and that there is an upper bound on transaction validation time λ, the R-SWAP protocol satisfies the safety property of a time locked atomic cross-chain swap.*

Proof. Consider an atomic cross-chain swap executed via the R-SWAP protocol and assume the safety of the transaction has been violated. This safety violation implies that there exist two smart contracts SC_1 and SC_2 one being in state *Redeemed* and the other *Refunded*. Indeed, given the graph in Figure 4 appendix B, any other set of state is impossible or doesn't imply a safety violation. We also exclude the case of an asset being locked forever because of the time bounded termination property.

SC_1 being redeemed and SC_2 refunded, imply that Bob has found the secret s because SC_1 is programmed to change state to redeemed only if provided s. This lead to a contradiction because it is impossible to calculate s from h.

SC_1 being redeemed and SC_2 being refunded imply that Alice has called the redeem function of SC_2. By doing so she has revealed the secret s. But given that the adapter checks the state of SC_2 at every new block, and given that the block time is $<< \Delta_2$, at the time Alice will be allowed to call refund, SC_1 will already be in state *Redeemed*, leading to a contradiction. □

Lemma V.3. *Assuming Bob's adapter is correct and that there is an upper bound on transaction validation time λ, the R-SWAP protocol satisfies the liveness property of an atomic cross-chain with high probability.*

Proof. Consider a time locked atomic cross-chain swap executed via the R-SWAP protocol and assume there is a violation of the liveness property. A violation of the liveness property implies that upon lock time expiration both contract are in state *Redeemed* but one or more of the participant have not received their payoffs. Since SC_1 and SC_2 are programmed to transfer the assets prior to transition to state *Redeemed*, a violation of the liveness property implies that the receiver address provided during the commit transaction $tx1_{com}$ of SC_1 resp. $tx2_{com}$ of SC_2 was not Bob resp. Alice.

When calling SC_1 COMMIT function, Bob's adapter has provided the details of the PRECOMMIT transaction $tx2_{pre}$ of SC_2. Then the COMMIT function of SC_1 has called $R_{a\leftarrow b}$ to verify $tx2_{pre}$. If the transaction is valid, SC_1 proceed to extract Bob's address from $tx2_{pre}$, set him as receiver and changes its state to *Committed*. Yet a violation of the liveness property would suppose that this address was something else, leading to a contradiction.

SC_2 COMMIT function works the same way, but instead of providing the PRECOMMIT transaction, Alice's adapter provides the COMMIT transaction details $tx1_{com}$. Prior to committing, SC_2 calls $R_{b\leftarrow a}$ to verify $tx1_{com}$. If it was a valid transaction, SC_2 extracts Alice's address from $tx1_{com}$ and calls its own COMMIT function with the address previously extracted as a parameter. Yet a violation of the liveness property would suppose that the address provided was something else, leading to a contradiction. □

*a) **Note on non-deterministic blockchains**:* With some proof-of-work blockchain systems such as Bitcoin and Ethereum, block creation is not deterministic but probabilistic. However it is proven in [19] that under the assumption of an honest majority of nodes $t \leq \frac{1-\delta}{2}(n-t)$, Bitcoin satisfies agreement and validity properties with probability at least $1-e^{-\Omega(\epsilon^2\lambda f)}$ with n number of parties mining; t out of which are controlled by the adversary, δ the advantage of honest parties, ϵ the quality of concentration of random variables in typical executions, λ the tail-bounds parameter and f the probability at least one honest party succeeds in finding a POW in a round. Thus, for such blockchain systems, R-SWAP satisfies the properties of a time locked atomic cross chain swap with high probability.

VI. PROTOCOL EVALUATION

In this section we will analytically evaluate the performances of the R-SWAP protocol. Then we will estimate the operational cost of maintaining the infrastructure required. A numerical projection is given in appendix.

A. Atomic Swap Latency definition

Naively, Atomic swap latency could be defined as the time difference between the moment first asset is locked and the moment the last asset is unlocked. But since there is no reliable global clock in most blockchain systems, it is difficult to measure the real latency of an atomic cross-chain swap. In order to be more accurate, latency can be measured from the participants blockchain client's perspective. Besides the improved accuracy, the measurement of the latency will better reflect the protocol ergonomics and performance.

Definition 3 (Atomic Swap Latency) Let $Locked$, $Refund$ and $Redeem$ be three predicates. Let p be a participant in an atomic cross-chain swap. Predicate $Locked = true$ indicates that p's asset is locked. Predicate $Refund = true$ indicates that p has received his refund and can transact it. Predicate $Redeem = true$ indicates that p has received his payoff and can transact it. Let t_l be the time when $Locked = true$. Let t_u be the time when $Refund \vee Redeem = true$. Then the atomic swap latency is defined by:

$$\Delta_{latency} = t_u - t_l \tag{10}$$

B. R-SWAP Latency

In the following, we are going to analyse the latency of the R-SWAP protocol. We will evaluate the theoretical latency of the protocol but we provide a numerical analysis in Appendix C.

Disambiguation: We refer to validation time as the time it takes for a transaction to be included in a block and to receive k confirmations. We refer to confirmation time as the time it takes for a transaction to be included in a block.

*1) **Theoretical analysis**:* λ is the upper bound on transaction validation with probability ϵ and Δ is the time lock value of a R-SWAP contract SC. There are two values for the latency, one from Alice's client perspective and one from Bob's. As proven in Section V, the R-SWAP protocol is γ time bounded. This means that from the moment Alice's COMMIT transaction has been sent and the moment she will have received her payoff or will be able to be refunded, there is a maximum of γ time. In the case every participant is honest and abide by the protocol, the atomic swap latency of the R-SWAP protocol is less than Δ_1, the lock time of contract SC_1. In the case Bob stopped abiding by the protocol after Alice's COMMIT transaction, she will have to wait for Δ_1's expiration before calling REFUND. Then this REFUND transaction will be confirmed in less than λ_a, the transaction validation time of blockchain BC_a. From Alice's point of view, this is a latency of less than $\Delta_1 + \lambda_a$. In the case Alice stopped abiding by the protocol after Bob's COMMIT transaction, he will have to wait for the expiration of Δ_2 before calling REFUND on SC_2. This gives a latency of $\Delta_2 + \lambda_b$ for Bob and $\Delta_1 + \lambda_a$ for Alice. Thus the overall maximum latency is $\Delta_l antency = 3\lambda_b + \lambda_a$.

Case \ Max latency	Alice	Bob
Alice $ Bob abide by the protocol	Δ_1	Δ_1
Bob stops after Alice's Commit	$\Delta_1 + \lambda_a$	None
Alice stops after Bob's Commit	$\Delta_1 + \lambda_a$	$\Delta_2 + \lambda_b$

Fig. 2. Table of R-SWAP theoretical latency

C. Cost evaluation

In this section, we will analyse the operational cost of the infrastructure required to implement the R-SWAP protocol. The R-SWAP protocol is made out of three software bricks; hash time locked contracts, blockchain adapters and relays. Operating an external adapter is as costly as operating a blockchain node. A single hash time locked contract can be used for a large amount of swaps, making the deployment cost negligible. Thus the operational costs of the infrastructure narrows down to the cost of operating the relays.

*a) **Operational cost of Relays**:* As explained in Section III-B0b, a relay $R_{a \leftarrow b}$ is a piece of software, possibly a smart contract, that allows to read and verify the state of chain BC_b from chain BC_a [20]. The intuition is that a relayer r publishes every new block header of chain BC_b to the relay $R_{a \leftarrow b}$ on chain BC_a. The relay implements the block verification algorithm of chain BC_b. It verifies the proof of work for PoW chains or the signature of $2/3$ of the validators for BFT consensus chains [21]. Then, once the block has been verified, a process can call the relay to verify a specific transaction of chain BC_b from BC_a. He pays a little fee that will compensate the relayer's work and expenses. Depending on the blockchain, this process can be both calculation and storage intensive. Unfortunately, storage and calculation "on-chain" is expensive in most blockchain systems, inevitably leading to high operational costs. Current implementation such as BTCRelay [10] used about 194 000 gas to store and verify a single Bitcoin block header [1]. Gas price and ether price are subject to high volatility. Thus it is important to note that what follows can vary from one day to another but to give a rough estimate, as of today's prices [2], this would translate to 25$ per relayed block. Considering that an average of 144 Bitcoin blocks are mined every day, maintaining the BTCRelay would cost alone about 3600$/day. If the volume of R-SWAP transactions is large enough, those operational costs could be amortized by an economy of scale, each swap participant paying a small fee to the relayer. The other outlays concerning the relays is the computational power required to verify a submitted transaction. For each R-SWAP execution each participants request at least one transaction verification to the relay. On modern implementation of BTCRealy, the cost of such an operation vary from 67000 to 102000 gas [18].

As of today, the operating cost of the relay is quite dissuasive, especially considering that current centralized exchange platforms offer fees as low as 0.1% [3]. Fortunately recent research suggest that the operational cost of blockchain relays could be reduced of up to 92% [21], thanks to an "on-demand" approach.

VII. Conclusions and Discussions

This paper presents R-SWAP, a time bounded atomic cross-chain swap protocol that makes use of blockchain relays and adapters to address the shortcomings of the hash locking technique. Previously proposed atomic cross-chain swap protocols such as [5] were subject to potential safety violation. Indeed, the safety of such protocols could only be satisfied with the assumption that no errors occurred on the participant's blockchain client. Moreover, the smart contract auditing tasks were to be executed by the participants, leading to bad ergonomics and potential safety violations. The R-SWAP protocol makes use of blockchain relay $R_{a \leftarrow b}$ to verify transactions of chain BC_b from chain BC_a thus automating the contract auditing process. The protocol also involves distributed blockchain APIs, namely blockchain adapters that reduces the risks of a client crash compromising the swap's safety.

We formally prove that R-SWAP protocol satisfies the properties of a time bounded atomic cross-chain swap with a maximum latency $\Delta_{latency} = 3\lambda_b + \lambda_a$, where λ's represents

[1] https://etherscan.io/address/0x41f274c0023f83391de4e0733c609df5a124c3d4

[2] eth price: 1293$, gas price: 100Gwei

[3] https://www.binance.com/en/fee/schedule

a specific blockchain's upper bound on transaction validation time with high probability.

The operational cost of the infrastructure required for the R-SWAP protocol can be narrowed down to the cost of the relay. Indeed the blockchain relay is undoubtedly the piece of software that has the highest operational cost, as high as 3600$ per day for a Bitcoin relay on the Ethereum blockchain at current price. However recent research [21] suggest that operational cost of relays could be reduced by 92%.

REFERENCES

[1] N. Satoshi, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[2] N. Ton, "A complete list of cryptocurrency exchange hacks [updated]," Jul 2020. [Online]. Available: https://blog.idex.io/all-posts/a-complete-list-of-cryptocurrency-exchange-hacks-updated

[3] J. Xu and B. Livshits, "The anatomy of a cryptocurrency pump-and-dump scheme," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1609–1625.

[4] B. Vitalik, "Chain interoperability," 2016.

[5] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, ser. PODC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 245–254. [Online]. Available: https://doi.org/10.1145/3212734.3212736

[6] V. Zakhary, D. Agrawal, and A. El Abbadi, "Atomic commitment across blockchains," *Proceedings of the VLDB Endowment*, vol. 13, no. 9, 2020.

[7] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 193–210.

[8] R. Han, H. Lin, and J. Yu, "On the optionality and fairness of atomic swaps," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 62–75.

[9] Z. D. Kiayias Aggelos, "Proof-of-work sidechains," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 21–34.

[10] Consensys, "ethereum/btcrelay," Oct 2017. [Online]. Available: https://github.com/ethereum/btcrelay

[11] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, 2014.

[12] J. Redman, "Report: Blockchain price oracle manipulation produces millions in losses, shows no signs of slowing – altcoins bitcoin news," Nov 2020. [Online]. Available: https://news.bitcoin.com/report-blockchain-price-oracle-manipulation-produces-millions-in-losses-shows-no-signs-of-slowing/

[13] S. Jon, "Chainlink exploits lead to eth losses-again," Sep 2020. [Online]. Available: https://coingeek.com/chainlink-exploits-lead-to-eth-losses-again/

[14] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, p. 228–234, Apr. 1980. [Online]. Available: https://doi.org/10.1145/322186.322188

[15] E. Anceaume, R. Ludinard, M. Potop-Butucaru, and F. Tronel, "Bitcoin a Distributed Shared Register," in *SSS 2017 - 19th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, ser. Lecture Notes in Computer Science, vol. 10616. Boston, MA, United States: Springer, Nov. 2017, pp. 456–468. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01522360

[16] Y. Amoussou-Guenou, A. D. Pozzo, M. Potop-Butucaru, and S. Tucci Piergiovanni, "Dissecting tendermint," *CoRR*, vol. abs/1809.09858, 2018. [Online]. Available: http://arxiv.org/abs/1809.09858

[17] C. Patrick, "Building and using external adapters," Jan 2021. [Online]. Available: https://blog.chain.link/build-and-use-external-adapters/

[18] Interlay, "interlay/btc-relay-solidity," 2020. [Online]. Available: https://github.com/interlay/BTC-Relay-Solidity

[19] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 281–310.

[20] A. Zamyatin, "Re-implementing btc relay in solidity," Jan 2019. [Online]. Available: https://www.alexeizamyatin.me/reimplementing-btcrelay-in-solidity/

[21] P. Frauenthaler, M. Sigwart, C. Spanring, and S. Schulte, "Testimonium: A cost-efficient blockchain relay," *arXiv preprint arXiv:2002.12837*, 2020.

[22] CoinMarketCap, "Cryptocurrency prices, charts and market capitalizations." [Online]. Available: https://coinmarketcap.com/

[23] D. Koops, "Predicting the confirmation time of bitcoin transactions," *arXiv preprint arXiv:1809.10596*, 2018.

[24] S. Kasahara and J. Kawahara, "Effect of bitcoin fee on transaction-confirmation process," *arXiv preprint arXiv:1604.00103*, 2016.

[25] J. Newbery, "An introduction to bitcoin core fee estimation," Oct 2018. [Online]. Available: https://bitcointechtalk.com/an-introduction-to-bitcoin-core-fee-estimation-27920880ad0

[26] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*, 2013, pp. 1–10.

[27] G. A. Pierro and H. Rocha, "The influence factors on ethereum transaction fees," in *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2019, pp. 24–31.

[28] E. G. Station. (2021) Eth gas station. ethgasstation.info. [Online]. Available: https://ethgasstation.info/

[29] P. Siriwardena, "The mystery behind block time," Jul 2018. [Online]. Available: https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a

[30] Rolandkofler, "rolandkofler/blocktime," 2017. [Online]. Available: https://github.com/rolandkofler/blocktime

[31] J. Kwon and E. Buchman, "Cosmos: a network of distributed ledgers (2016)," 2016. [Online]. Available: https://cosmos. network/whitepaper

[32] S. Braithwaite, E. Buchman, I. Konnov, Z. Milosevic, I. Stoilkovska, J. Widder, and A. Zamfir, "Tendermint blockchain synchronization: Formal specification and model checking," in *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, T. Margaria and B. Steffen, Eds. Cham: Springer International Publishing, 2020, pp. 471–488.

[33] F. Omar, "Cosmos network now has nearly 100 validators, 6-7 second block times," Apr 2019. [Online]. Available: https://www.cryptoglobe.com/latest/2019/04/cosmos-network-now-has-nearly-100-validators-6-7-second-block-times/

[34] G. Birch, "Cosmos stargate update overview," 2020. [Online]. Available: https://figment.io/resources/cosmos-stargate-upgrade-overview/#ibc

[35] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework." [Online]. Available: https://github.com/polkadot-io/polkadotpaper/raw/master/PolkaDotPaper.pdf

[36] T. Nolan, "Alt chains and atomic transfers," 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949

[37] M. Belotti, S. Moretti, M. Potop-Butucaru, and S. Secci, "Game theoretical analysis of atomic cross-chain swaps," in *40th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[38] K. team, 2018. [Online]. Available: https://github.com/KomodoPlatform/BarterDEX

[39] K. Platform, "Komodo, an advanced blockchain technology, focused on freedom," June 2018. [Online]. Available: https://static2.coinpaprika.com/storage/cdn/whitepapers/140811.pdf

[40] Blockchain.io, "Blockchain.io your gateway to the internet of value." [Online]. Available: https://blockchain.io/

APPENDIX A
SMART CONTRACT STATES REPRESENTATION

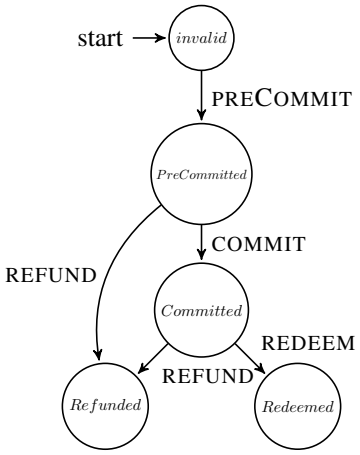

Fig. 3. Representation of the R-SWAP smart contract states as a Directed Acyclic Graph

APPENDIX B
PROTOCOL REPRESENTATION

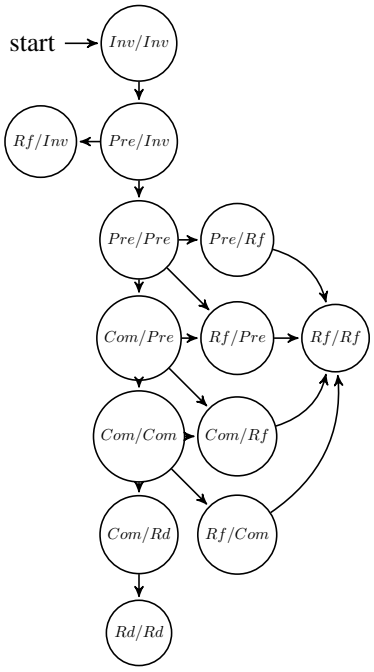

Fig. 4. Representation of the R-SWAP protocol possible set of states as a Directed Acyclic Graph

This directed acyclic graph represents the states of both contracts on each chain. Inv for Invalid, Pre for PreCommitted, Com for Committed, Rd for Redeemed, Rf for Refunded.

APPENDIX C
NUMERICAL ANALYSIS OF R-SWAP LATENCY

The value of the latency is highly dependant on the set of blockchain involved in the swap, because time locks are based on the upper bound of transaction validation time λ with probability ϵ, which is specific to each blockchain. In this section we will provide a numerical analysis of the swap latency between two most valued cryptocurrencies in terms of market cap at the time of writing: Bitcoin and Ethereum [22].

*a) **Bitcoin's upper bound on transaction validation time**:* Calculating Bitcoin's or any blockchain's upper bound on transaction validation time is a very complex problem. Indeed, it depends on numerous factors such as transaction fees, network traffic, current hashrate and difficulty, size of the unconfirmed transaction mempool etc. But it has been shown in [23] that the main factors are transaction fee density (in satoshi/Byte) and network traffic.

It is known that miners order transactions in their mempool by fee density to maximize profitability [24]. By setting sufficiently high fee density it is possible to predict that a transaction will be included in the next block with a probability p as high as 95% [25]. Then to find an estimation of Bitcoin's upper bound on transaction validation time, we must ensure that once the transaction has been included, there is enough time for $k = 6$ new blocks to be appended to the chain, i.e., to receive $k = 6$ confirmations. Block time in Bitcoin follows an exponential distribution with parameter $\theta = 0.001578$ [26]. Thus the probability of k blocks being mined in less than x time is given by the PDF of the gamma distribution Gamma(k,θ) cumulative function.

As most blockchain clients consider $k = 6$ being a safe number of confirmations we obtain the value of Figure C-0a.

$P(X < x)$	x (in seconds)
0.95	6662.25
0.99	8307.02
0.995	8966.89
0.999	10427.60

Fig. 5. Probability for 6 blocks to be mined in less than x seconds

Thus with a fee density high enough for the transaction to be included in the next block with high probability $p = 0.95$, there is an upper bound on this transaction validation time $\lambda_{btc} = 10427$ seconds with high probability. [4] Rounded up, this translates to a 3 hours validation time.

*b) **Ethereum's upper bound on transaction validation time**:* Because Ethereum also uses proof of work, the factors influencing transaction validation time are similar to the ones with Bitcoin [27]. As in Bitcoin, miners also tend to order transaction depending on the transaction fees (named gas price in Ethereum). Services such as Ethereum Gas Station [28] provide information on gas price relative to confirmation time. For each target confirmation time, they provide ranges of gas prices. For instance the "fast" range of gas price would have a transaction confirmed in less than two minutes, while the

[4] $p = 0.95 * 0.999 = 0.949$. It is to be noted that even if the transaction has not received 6 confirmations yet, it should have received at least 5 confirmation with probability > 0.949, and should receive it's 6th confirmation in the next ten minutes

"standard" gas price in less than five. It has been shown in [27] that with the highest range of fees, it takes at most thirty seconds (or two blocks) to have a transaction confirmed.

Now that we now how much time it takes to have an Ethereum transaction included in a block, we need to find how much time does it take for this transaction to become k-valid, i.e., for k new blocks to be appended to the chain.

Most Ethereum clients consider that $k = 12$ is a safe number of confirmation. We could naively multiply the average block time by k to obtain the upper bound on transaction validation time for Ethereum, but unlike Bitcoin, Ethereum is not set to have a constant block time. Indeed, the difficulty adjustment algorithm is such that, at some point, the difficulty will rise exponentially, "freezing" the blockchain. This event, referred to as the "Ice Age" has been thought of to force miners to switch to Proof of Stake [29]. Thus the estimation we will provide for Ethereum's upper bound on transaction validation time only holds prior to this event.

A study based on the Ethereum blockchain data [30] showed that 99% of the blocks where produced in less than a minute. Thus with $k = 12$, twelve minutes after its inclusion in a block, a transaction should be k-valid with high probability $p \approx 1$. Therefore with sufficient gas price, there is an upper bound on transaction validation time for Ethereum $\lambda_{eth} = (k * 60) + 30 = 750$ seconds with high probability. [5]

*c) **Latency for a swap between Bitcoin and Ethereum**:* Consider a R-SWAP protocol execution between Bob and Alice. Alice, the initiator, has bitcoins. Bob, the participant, has ethers. (The initiator is the one that generates the secret). Given the values of upper bounds on transaction validation time, given the theoretical latency values of Figure VI-B1 and given the values of time locks of Section V-A, the estimation of the maximum latency values for this swap are presented in Figure C-0c. (We selected $\Delta_{btc} = 8307$ seconds and $\Delta_{eth} = 750$ seconds).

Case \ Max latency	Alice	Bob
All participants abide by the protocol	10 557	10 557
Bob stops after Alice's Commit	18 864	n.a
Alice stops after Bob's Commit	18 864	1500

Fig. 6. Latency for a R-SWAP execution between Bitcoin and Ethereum

*d) **Tendermint upper bound on transaction validation time**:* Block-chain systems such as Cosmos [31] uses the Tendermint byzantine fault tolerant (BFT) consensus [32]. Tendermint BFT consensus satisfies the instant finality property. With our model, this translates to having a safety factor $k = 1$. Thus, the Tendermint upper bound on transaction validation time is nothing else than Tendermint block time. As of today, the observed block time is seven seconds [33]. Thus for Tendermint we have $\lambda_{tendermint} = 7$.

*e) **Latency for a swap between Tendermint and Ethereum**:* Consider a R-SWAP protocol execution between Tendermint and Ethereum. The initiator (the one generating the hash lock) has ethers and the participant has atoms (Cosmos native asset). For such a swap we obtain the latency values presented in Figure C-0e. Cosmos recently launched the Stargate update [34] which allows cross chain transactions. The inter blockchain communication protocol used to achieve cross chain transactions uses some sort of pegging like X-Claim [7]. As this is a work in progress, we do not have a performance evaluation of this protocol.

[5]This is a 12min and 30s total validation time.

Case \ Max latency	Alice	Bob
Alice $ Bob abide by the protocol	771	771
Bob stops after Alice's Commit	1521	n.a
Alice stops after Bob's Commit	1521	14

Fig. 7. Latency for a R-SWAP execution between Ethereum and Tendermint

APPENDIX D
OTHER RELATED WORKS

There are currently several operational systems for achieving interoperability between different blockchains (e.g Cosmos [31] or Polkadot [35]). However, they are not yet fully formalized and proved correct. Moreover, there is no academic study focusing on their performances.

Blockchain interoperability protocols can be classified into two main classes according to their level of decentralization: systems that use a trusted third-party to validate transactions and systems that realize it directly between blockchains without the need of a trusted third-party.

The first atomic swap was proposed for Bitcoin by Nolan [36]. This solution uses hash-time locked contracts enabling conditional assets transfers. In [5] the authors generalize Nolan's scheme and propose its analyses using a game theoretical approach. This analysis has been refined later in [37]. Other projects such as BartherDEX [38], part of the Komodo project [39], represents a cross-chain solution that matches orders and defines the swap protocol or Blockchain.io [40] implements atomic cross-chain swaps by combining centralized components (order matching) with decentralized ones (trade settlement and execution). These projects are not yet formally proved correct.

The academic research focuses on hybrid swap protocols, replacing decentralized commitment/locking schemes (hash-locks) with centralized ones, resulting in more attractive and efficient protocols. AC3TW and AC3WN [6] protocols propose atomic cross-chain swaps respectively with centralized and distributed trusted authorities (i.e., witnesses).

APPENDIX E
R-SWAP SMART CONTRACT

Algorithm 2 R-SWAP hash time lock contract class

address $sender$; ▷ Address of swap initiator
address $receiver$; ▷ Address of participant
float $amount$; ▷ Amount exchanged
float $rate$; ▷ Exchange rate
integer $timelock$;
bytes $secret$;
bytes $hashLock$; ▷ Hash of the secret
address $Relay$; ▷ Address of the relay smart contract
enum State{Invalid, PreCommitted, Committed, Redeemed, Refunded}
State $state \leftarrow Invalid$ ▷ Variable to store state
procedure PRECOMMIT($hashLock$, $amount$, $rate$)
    requires($this.state = Invalid$)
    $this.hashLock \leftarrow hashLock$
    $this.amount \leftarrow amount$
    $this.rate \leftarrow rate$
    $this.sender \leftarrow msg.sender$
    $this.state \leftarrow PreCommitted$
end procedure
procedure COMMIT(tx)
    requires($this.state = PreCommitted$)
    requires(RELAY.VERIFYTX(tx)) ▷ Calling the relay to verify that the provided transaction is valid
    requires($tx.hashLock = this.hashLock$) ▷ Verifying the hash lock
    requires($tx.amount = this.amount * this.rate$) ▷ Verifying the amount
    $this.receiver \leftarrow msg.sender$ ▷ Setting the function caller as receiver
    $this.timelock \leftarrow now + \Delta_1$ ▷ Setting up the time lock
    $this.state \leftarrow Committed$
end procedure
procedure REDEEM($secret$)
    requires($this.state = Committed$)
    requires($msg.sender = this.receiver$) ▷ Check that the caller is the swap receiver
    requires(SHA256($secret$)= $this.hashLock$) ▷ Verify the secret
    $this.secret \leftarrow secret$ ▷ Make the secret public
    TRANSFER($this.amount$, $this.receiver$) ▷ Transfer the funds
    $this.state \leftarrow Redeemed$
end procedure
procedure REFUND
    requires($this.state \neq Redeemed$)
    requires($msg.sender = this.sender$) ▷ Verify that the caller is the swap sender
    requires($this.timelock \geq now$) ▷ Check that timelock is elapsed
    TRANSFER($this.amount$, $this.sender$) ▷ Transfer the funds
    $this.state \leftarrow Refunded$
end procedure
procedure CHECKSTATE
    return $this.state$
end procedure
procedure CHECKSECRETKEY
    return $this.secret$
end procedure

msg is a variable generated when executing a transaction. It contains several information relative to the transaction such as the address of the function caller $msg.sender$ or the value of the transaction $msg.value$.