h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

– Fachbereich Informatik –

Atomic Cross-Chain Swaps with Ethereum Blockchains

Abschlussarbeit zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von

Christopher Glißner

Matrikelnummer: 732978

Referent	:	Prof. Dr. Ronald Moore
Korreferent	:	Prof. Dr. Stefan Rapp

DECLARATION

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 28. April 2020

Christopher Glißner

ABSTRACT

Today's blockchain ecosystem has spread into multiple chains using different consensus algorithms and architectures. After a decade of blockchain applications and development there appears the need for interoperability between upcoming and current blockchain innovations. Interoperability presents a complex task towards a wider uptake of blockchain technologies to swap assets and tokens between chains in a steadily growing environment. This ecosystem consists of many chains without any interoperability. For example if Alice wants to trade her bitcoin with Bob's ether they have to use a centralized exchange. Which exposes Them to the possibility of scam, fraud and malicious hacks. Atomic Cross-Chain Swaps propose a small step towards interoperability of blockchains. Interoperability removes the intermediate parties such as exchanges. Blockchain interoperability enables more security for the end-user and closes attack vectors for fraudulent actors.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ABKÜRZUNGSVERZEICHNIS

CT Confidential Transactions

EVM Ethereum Virtual Machine

EVM Code Ethereum Virtual Machine Code

ISO International Organization For Standardization

PoW Proof-of-Work

PoS Proof-of-Stake

SHA-256 Secure Hash Algorithm 256

SHA Secure Hash Algorithm

Part I

THESIS

1 INTRODUCTION

This Chapter will discuss the motivation and several historic information which led to the research objective covered by this thesis. Next chapter will discuss the model and general principles serving as foundation to address the research objective. Finally the implementation of atomic cross-chain swaps on Ethereum blockchains will be presented. Followed by an outlook into the future of blockchain interoperability research will be an own chapter. The last chapter will be a conclusion of this work, to reflect the whole thesis.

1.1 MOTIVATION

Decentralization and scaling of blockchains is a widely addressed topic by many researchers. Bitcoin [29] was the first successful blockchain application which decentralizes money, combining the well-known concepts of Proof-of-Work (PoW) algorithms and the distributed ledger concept. Based on previous work by Wood [40], Buterin described the next-generation Smart Contract and decentralized application platform known as Ethereum [11] [12]. While most present blockchains still remain unconnected, pegged sidechains are formally defined by [3] and happens to be the foundation for many proposals towards blockchain interoperabilty. Herlihy proposed the concept of atomic cross-chain swaps to exchange tokens and assets between blockchains safely [19] to enable communication between two blockchains without intermediaries. This will raise security for the end-user. To address the blockchain ecosystem further and underline how diverse it is in terms of technology and architecture, this chapter will also give a quick introduction to two different Cryptocurrencies. The following are just two of a vast amount of cryptocurrencies and technologies out there[1]. Originally known as BitMonero, the Monero cryptocurrency was created in April 2014 [2] as a proof-of-concept currency of the CryptoNote [35] technology. Cryptocurrencies dont just exist with different consesus algortithms and architecures, they also aim to serve a specific purpose. Monero addresses privacy and utilitizes borromean ring signatures [27] to implement ring Confidential Transactions (CT) [30] in order to improve untraceability. Another good example to show how diverse the blockchain ecosystem is, is Cardano which is a decentralised public blockchain and cryptocurrency project. Cardano uses a research-first driven approach to develop a smart contract platform which seeks to deliver more advanced features than any protocol previously developed. For example the initiators developed their own Ouroboros Proof-of-Stake (PoS) protocol [22]. These few examples already point out the diversity of the blockchain ecosys-

1 **CoinMarketCap - Cryptocurrencies: 5, 553 - https://coinmarketcap.com/. Retrieved 8 June 2020.**

tem and underline why it is so important to address interoperability between blockchains in the future.

1.2 RESEARCH OBJECTIVES

This work covers three research objectives in general, which are equally weighted in importance. The chapter model and general principles will explain the basics like the ethereum blockchain and smart contracts to build the foundation to understand the implementation of atomic cross-chain swaps. Besides the Ethereum Virtual Machine (EVM) and smart contacts digraphs will also be discussed, since they are crucial to execute a secure swap between two or three blockchains. The second part will introduce the implementation of atomic cross-chain swaps between three blockchains (three-way swap) and will discuss the reference architecture and the implementation of the smart contracts in detail. The last part will be all about future research and an outlook about current blockchain research to show the direction and possible outcome of an blockchain ecosystem which we might see in the future. Since there is a vast amount of blockchains with different consensus algorithms and software architectures there seems to be a need for interoperability between these chains. Some researchers do even propose whole frameworks in order to remove the intermediate party and swap information, token or assets back and forth between these chains. Ultimately this work will provide an implementation of atomic cross-chain swaps between three ethereum blockchains, enabling end users to trade information, assets or token between three chains without an intermediate party and discuss an overview about current and future blockchain research addressing interoperability.

2

MODEL AND GENERAL PRINCIPLES

This chapter will discuss the model and general principles, which serve the foundation to understand the implementation reports of the following chapter. Its important to understand every subsection of this chapter, such as the ethereum blockchain (chapter 2.1.1), PoW consensus algorhithm (chapter 2.1.1), the EVM (chapter 2.1.2), smart contracts (chapter 2.1.3) and all topics needed for atmoic cross-chain swaps (chapter 2.2 and 2.3).

2.1 ETHEREUM BLOCKCHAIN

The Ethereum blockchain was introduced in Vitalik Buterin's paper in 2013, which addressed several limitations of the Bitcoin's scripting language [11] based on Buterin's white paper Wood released the yellow paper one year later [40]. The main contributions are full turing-completeness and saving all states of computations in between the states [17]. Through its own programming language Solidity, it provides an abstract layer enabling anyone to create their own rules for ownership, formats of transactions, and state transition functions [37]. Development was funded by an online crowdsale that took place between July and August 2014 [34]. The system then went live on 30 July 2015[1] and is called the ethereum public blockchain. Which means its accessible to everyone who wants to send Ether token or execute smart contracts. In general we have to distinguish between private and public blockchains. The public one is described above. Since everybody can go to github and fork e.g the geth client its possible to start an own private blockchain. In an enterprise software context, where corporate stakeholders are given certain rights and privileges to read and write to the company chain, the deployment is known as a permissioned blockchain. Nevertheless for both the private (permissioned) and the public blockchain its possible to do the following [17]:

- Send and receive Ether
- Write smart contracts
- Create provably fair applications
- Launch your own token based on Ether

1 **"Ethereum Launches" - https://blog.ethereum.org/2015/07/30/ethereum-launches/. Retrieved 30 July 2015.**

2.1.1 *Proof-of-Work Consensus Algorithm*

The PoW system of bitcoin and ethereum is similar to Adam Back's Hashcash [4], but instead of a newspaper or Usenet post, the PoW involves scanning for a value, which is hashed with the Secure Hash Algorithm 256 (SHA-256) to begin with a number of zero bits. The average work required is exponential. That means the more leading zeros, the more work. In the end the work can be later verified by executing a single hash. The PoW is implemented by incrementing a nonce in the block until a value is found that gives the block's hash the required leading zero bits. The majority decision is represented by the longest chain, which has the greatest PoW effort invested in it. Because later blocks are chained after the premature block, the work to change the block would include redoing all the blocks after it. To modify a past block, an attacker would need to redo all the past work until the specific block he wants to change [29].

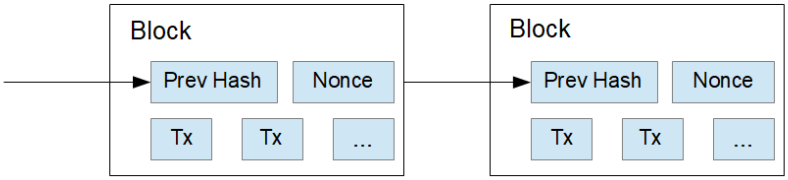

Figure 2.1: Blocks

Beginning with the genesis block, each single block is chained to the previous block by including the the previous hashes (figure 2.1). The SHA-256 takes a 256 bit length of input messages and hash them to fixed-length outputs [36]. So the SHA-256 function starts by padding the message according to the so-called Merkle-Damgård strengthening technique. Further the message is processed block by block with the the underlying compression function, which initializes an appropriate number of chaining variables to a fixed value to hash the genesis block and also the current hash value for the following blocks [15]. In the year 2003 the Secure Hash Algorithm (SHA) was published by the International Organization For Standardization (ISO) making SHA a standard for most countries in the world [1]. Five years later Satoshi Nakamoto proposed the first concept of a PoW based blockchain to allow for public agreement on the order of transactions. The public ethereum blockchain uses PoW to this day. There are several other consensus algorithms today suitable for a blockchain and some are considered to be an improvement to PoW, which will be discussed later in this work.

2.1.2 *Ethereum Virtual Machine*

This chapter will cover basics of the EVM and discuss the transaction execution to reach a general understanding what the EVM is and how it works. The EVM is a simple stack-based architecture. The word size of the machine is 256-bit and the stack has a maximum size of 1024 elements, meaning that the EVM can store a total of 1024 stack items with each 256-bit. The machine does not follow the standard von Neumann architecture. Rather than storing program code in generally-accessible memory or storage, it is stored separately in a virtual ROM. The EVM is different from the von Neumann architecure, because it can have exceptional execution thus as the out-of-gas exception [40]. The ethereum network can be seen as one single computational device, where the point of turning the machine on, is the creation of the genesis block. All blocks of a blockchain hold information about state changes of the EVM. The ethereum public blockchain for example holds all state transitions of the EVM since it was initially turned on. Since every state transition of the EVM is modelled as a transaction, a blockchain holds the whole history of states of the specific machine [17]. Before it will be put into a block each transaction is verified and validated before the next canonical block can be placed on the last one. Through the transaction validation, nodes on the network do not need to individually evaluate the trustworthiness of every single block in the network to compute the present balances of the accounts on the network. The client simply has to verify the hash of the parent block and see if the new block contains the correct hash its parent's transactions and state [17]. The current valid block in an ethereum network is known as world state (state), which is a mapping between addresses (160-bit identifiers) and account states. Thus being the current executional state of the virtual state machine, known as the EVM. It is considered as a quasi-touringcomplete virtual machine. Because the amount of total execution is intrinsically bound to the parameter gas, the quasi qualification is added [40]. The execution model is defined by the ethereum state transition function and specifies how system state is altered through bytecode executions.

The ethereum state transition function, APPLY(S, TX) -> S' is defined by Buterin [11] as follows:

1. Check for transaction and signature validity and check if the nonce of the sender account matches the receivers account's nonce. If not, return an error.

2. Calculate the transaction fee and determine the sending address from the signature. Increment the sender's nonce and substract the fee from the sender's account. Return an error, if the balance of the sender's account is lower than the fee.

3. Initialize GAS including a certain quantity of gas per byte to pay for the bytes in the transaction.

4. If the receiver's account doesnt exist, create it and then send the transaction value from sender's account to the receiver. If the receiver's account is a contract, run code execution until completed or gas is depleted.

5. In case the value transfer or code execution failed because theres not enough gas, revert all state changes except the payment of the fees and add the fees to the miner's account.

6. If code execution has completed, refund the fees for all remaining gas to the sender and also send the fees paid for gas consumed to the miner.

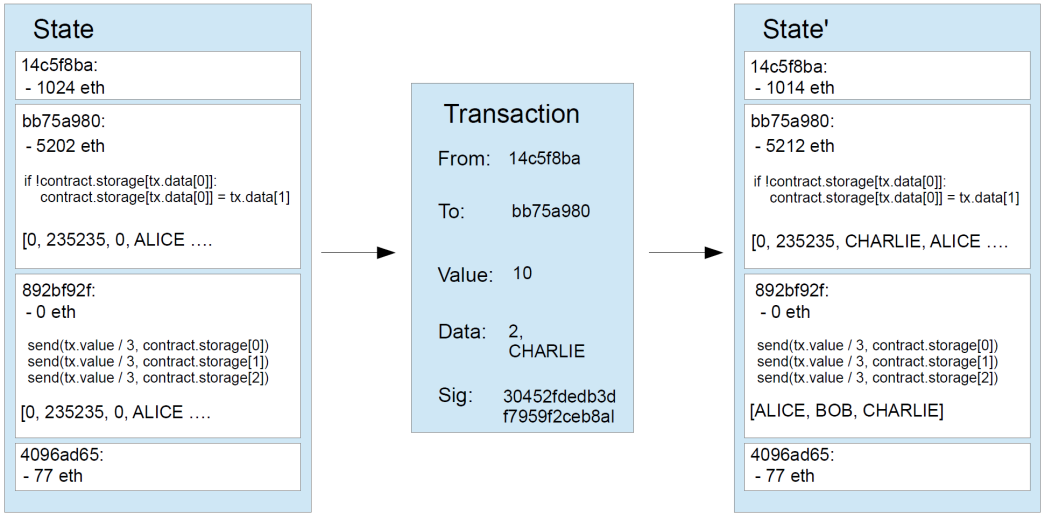

Figure 2.2: Ethereum State Transition Function

The Figure above shows all calculations applying the function APPLY(S, TX) -> S' to an example state (figure 2.2). Due to the distributed nature of the EVM and the fact its built of many nodes around the world to ensure distributed security, it must be purpose-built to solve the diffmatching problem that can occur, when there are many near-simultaneous changes to the same database from many actors in the system all around the world[2]. Solving this problem in a verifiable and trustworthy way is basically what the EVM does. It's resilience and security does increase with the amount of machines in the network and is implemented and backed by the gas fee system. The opportunity to earn gas as a miner is one reason for many nodes to participate in the network and at the same time raises security [17].

2 **Google Code "Diff-Match Patch" - https://code.google.com/p/google-diff-match-patch/ 2016.**

2.1.3 *Smart Contracts*

Around the 1990s it became clear that algorithmic enforcement of agreements could become common in the way humans cooperate. Early work towards smart contracts was done by Szabo [33] and Mark Miller [28]. Even though no specific system was proposed by them, they proposed that the future of law would be driven by such systems. Ethereum can be seen as such a system, it delivers a general implementation to execute financial contracts by a machine. The term smart contract was first introduced by Buterin [11], in scientific context they are simply called contracts by Wood [40]. In this context, contract refers to a specific kind of contract: a financial contract, also known as derivative or option. These contracts are agreements between two parties to buy or sell a specific asset at some point in the future or when certain conditions are met, like a specified price. In ethereum context a contract does basically the same thing as a financial contract. The contract is written as code and is an agreement between accounts, to send a payment to a specified address, when certain conditions are met. The reason these contracts are called smart contracts, is because they are executed by a machine, also known as the EVM. When certain conditions are met, the assets (ether or token) will be moved automatically by the machine. Assuming the system is still running, these smart contracts can also take effect hundreds of years after being deployed. As long as the EVM is running deployed contract maintain their validity. Its basically impossible for a party to back out of these contracts, because the smart contracts are empowered to move or hold assets and move them to specific accounts when the implemented conditions are met [17]. While smart contract are often written in the high level programming language solidity, the code the EVM understands and executes is written in a low-level, stack based bytecode language, referred to as Ethereum Virtual Machine Code (EVM Code). In general the bytecode consists of a series of bytes, where each byte represents an operation to be executed by the EVM and code execution is an infinite loop executing code and storing data to the following operations until an error or STOP or RETURN is reached [11]:

- Stack (last-in-first-out container to which 32-byte values are pushed or popped)
- Memory (infinitely expandable byte array)
- Storage (of a contract, changes here don't reset after computation ends they are persistent)

2.2 DIGRAPHS

A directed graph (digraph) $\mathcal{D}$ (figure 2.3) in the example consists of a finite nonempty set of objects called vertices. So the vertex set of the example 2.3 is $V(\mathcal{D}) = \{u, v, w, x, y, z\}$. It also has a finite set of A($\mathcal{D}$) of ordered pairs of distinct vertices called arcs (or directed edges according to [14]), in this example 2.3 $A(\mathcal{D}) = \{(u, v), (u, w), (w, u), (z, u), (x, z), (y, z)\}$. So the digraph $\mathcal{D}$ consists of $V(\mathcal{D})$ the vertex set and $A(\mathcal{D})$ the arc set of $\mathcal{D}$, which can also be expressed as $\mathcal{D} = (V, A)$.

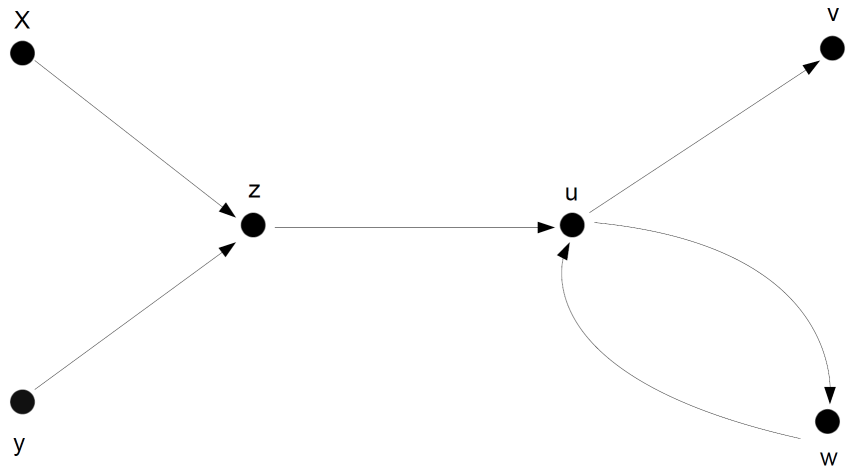

Figure 2.3: A Digraph D

The order (size) is sometimes denoted as $|\mathcal{D}|$, so the size of the digraph $\mathcal{D}$ in the example 2.3 is $|\mathcal{D}| = 6$. For an arc (u, v) the first vertex u is its tail and the second vertex v is its head, we can also say that the arc (u, v) leaves u and enters v. Another expression is, assuming (u, v) is an arc, one can say that u dominates v (or v is dominated by u) and can be denoted by $u \to v$. A digraph $\mathcal{D}$ is strongly connected (or just strong) if each vertex of $\mathcal{D}$ is reachable from every other of $\mathcal{D}$'s vertices. If its not possible to loop through a digraph $\mathcal{D}$, then it has no cycle and the digraph $\mathcal{D}$ is acyclic. The set S of vertices (arcs) is called a feedback vertex set or an feedback arc set, if $\mathcal{D}$ happens to be acyclic. Let $\mathcal{H}$ be another digraph, if $V(\mathcal{H}) \subseteq V(D)$, $A(\mathcal{H}) \subseteq A(D)$ and every arc in $A(\mathcal{H})$ has both end-vertices in $V(\mathcal{H})$, then H is a subdigraph of digraph $\mathcal{D}$. The distance from x to z is the minimum length of a (x, y)-walk and is denoted as $dist(x, y)$ of the given digraph $\mathcal{D}$ in figure 2.3. If a digraph $\mathcal{D}$ is strongly connected and only if $\mathcal{D}$ is strong, then the diameter of $\mathcal{D}$ is $diam(\mathcal{D}) = dist(V, V)$ [5]. The example in this chapter and its figure 2.3 is important to understand and to know the basic terminology, since the processes of atomic cross-chain swaps itself is based on digraphs.

2.3 ATOMIC CROSS-CHAIN SWAPS

An atomic cross-chain swap is like the name indicates an atomic transaction to swap assets, which is also across multiple blockchains. Altough ethereum can be seen as one system, its still a distributed ledger where state transitions are processed by all miners in the network, so the atomic cross-chain swap is a distributed coordination task (special case of distributed atomic transaction) [38]. Since the swap is atomic it either finishes the protocol or initiate refunds in case any party doesn't follow through the whole protocol. The concept of atomic cross-chain swaps is well known to the blockchain community and will be discussed later on in Chapter 3 (Related Work). All following concepts and models discussed are based on Herlihy's work [19]. A cross-chain swap is modeled as a directed graph $\mathcal{D}$, the vertexes of the digraph represent the parties and its arcs are proposed asset transfers. For any pair($\mathcal{D}$,L), where $\mathcal{D} = (V,A)$ is a strongly-connected directed graph and $L \subset A$ a feedback vertex set for $\mathcal{D}$. The three-way swap in specific has to be an acyclic and utilitizes a slightly modified form of a digraph introducing followers and leaders and using a form of hashed timelock contracts, read more in Section 2.3.2. The concept of atomic cross-chain swaps removes the intermediate party, which is usually an exchange marketplace for cryptocurrencies to trade assets or tokens. This chapter starts with an example of a three-way swap (Section 2.3.1), discussing the model of hashlocks and hashkeys (Section 2.3.2) and at last propose a general protocol (Section 2.3.3) for swaps.

2.3.1 *Example*

Lets assume Carol lifes in a state where car titles are managed on a blockchain and wants to sell her car for bitcoins. Alice is interested in Carol's car, but wants to pay in another cryptocurreny(following called alt-coin). Lucky for both that Bob is willing to trade his alt-coins for bitcoins. Since none of the parties trust each other, they can arrange a three-way swap to assure their transaction is atomic across the different blockchains. To complete the transaction Alice has to send her alt-coins to Bob, so he can transfer his bitcoin to Carol who will finally give the car to Alice. If all parties stick to the deal then everything will be fine, to assure that no one looses their funds due to a malicious actor who behaves irrationally this chapter and the following will discuss the protocol of atomic cross-chain swaps by the example of Alice, Bob and Carol's transaction. Today many blockchains come with smart contracts to execute a transfer of funds if certain conditions are met. Let $H(\cdot)$ be a cryptographic hash function. Alice could transfer her alt-coins to a smart contract which acts as escrow on the alt-coin blockchain, the contract also holds the information of hashlock h and timelock t. The hashlock h is determined by a contract value s, called a secret and let $h = H(s)$. If Alice sends the secret to Bob it allows him to claim the funds sitting in the smart contract's escrow using hashlock h. Timelock t means that Bob also needs to

produce a secret before the time t elapsed, if he does not manage to produce the secret in time, then the funds on the alt-coin blockchain are transferred back to Alice by the smart contract. Let Δ be enough time for the parties to publish their smart contracts or detect changes. Since Alice, Bob and Carol need to execute a three-way swap illustrated in figures 2.4 and 2.5 they need to apply to the following simple protocol:

- Alice publishes a smart contract on the alt-coin blockchain including a secret s, $h = H(s)$ she created. The contract also holds the information of hashlock h and timelock 6Δ in the future, to send her alt-coins to Bob.
- As soon as Alice's contract is confirmed on the alt-coin blockchain and detected by Bob, he deploys a smart contract on the bitcoin blockchain to transfer his funds to Alice. Bob adds the same hashlock h value to the contract like Alice and a timelock 5Δ in the future.
- When Carol detects Bob's confirmed smart contract on the bitcoin blockchain, she publishes a smart contract on the car title blockchain in order to transfer the title of her car to Alice. Again with same hashlock h, but timelock 4Δ in the future
- After Alice confirms Bob's mined contract on the title blockchain, she sends s to Carol's contract. Alice aquires the title and reveals s to Carol.
- Carol then sends s to Bob's contract, she aquires the bitcoin and reveals s to Bob.
- Bob completes the swap by sending s to Alice's contract and claiming the alt-coins.

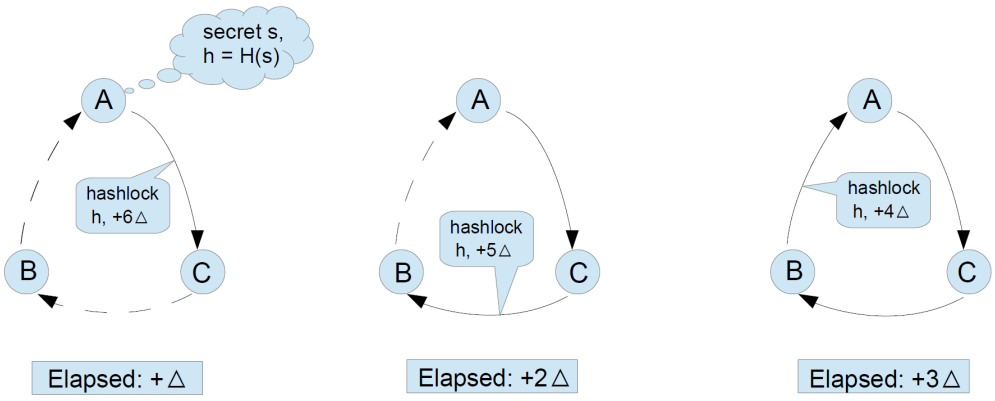

Figure 2.4: Atomic cross-chain swap: deploying contracts

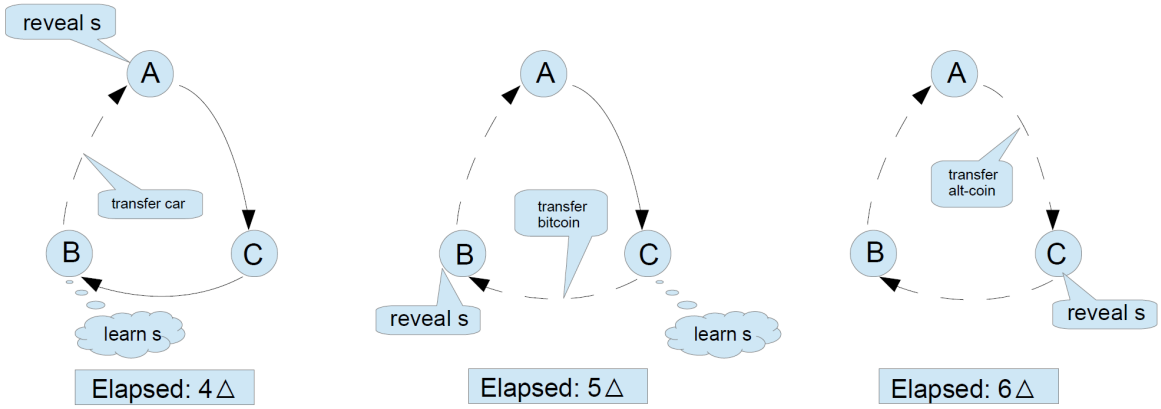

Figure 2.5: Atomic cross-chain swap: triggering arcs

So what could possibly go wrong in this three-way swap? If any party fails to publish their contract during deployment phase, then all contracts will timeout and trigger refunds of the assets. If any party fails while triggering arcs, then only that party will end off worse. For example, if Carol halts execution of the swap without triggering her contract, then Alice gets the car and Bob gets a refund. So in that case Carol would only harm herself. Also the order of contract's deployment matters. Because if Carol posts her contract with Alice, before Bob posts his contract with Carol, then Alice could claim the car title without paying Carol. The timelock values are also important. For example, if Carol's contract with Bob and Bob's contract with Alice would expire at the same time, then Carol would reveal s to collect Bob's bitcoin at the very last moment, leaving Bob no time to claim his alt-coins from Alice. If parties behave irrationally, then Alice might end up not getting her car. This would be the case if Alice (irrationally) reveals s before phase completion, then Bob can take Alice's alt-coin and Carol might be able to take Bob's bitcoin.

2.3.2 *Hashlocks and Hashkeys*

In a simple two-party swap each participating party publishes a contract that assumes temporary control of the assets. This hashed timelock contract[3] stores a pair (h, t) and holds a secret s. This contract triggers and releases funds if $h = H(s)$ is available before time t has elapsed. If the contract does not receive the secret before time t is elapsed, then the assets are refunded and the trade is cancelled. The example discussed earlier is a three-way swap, where each arc holds a single hashlock h and timelock t. Timeouts are assigned in a way that each entering arc of a follower v ensures that the timeout for the leaving arc is at least Δ ahead of the arc entering v. This gap ensures that if any of the digraph's arcs leaving v is triggered so v has enough time to trigger every entering arc. For example, if a cross-chain swap digraph has only one leader $\hat{v}$, then the follower's subdigraph is acyclic. In

3 **bitcoinwiki "Hashed timelock contracts" - https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts. 2020.**

the case of a three-way swap, the one that was discussed earlier in the example subsection, the timeout on arc(u, v) can be $(diam(\mathcal{D}) + D(v, \hat{v}) + 1) \cdot \Delta$. The lefthand side of figure 2.6 illustrates the length of the longest path $D(v, \hat{v})$ from v to the unique leader $\hat{v}$.

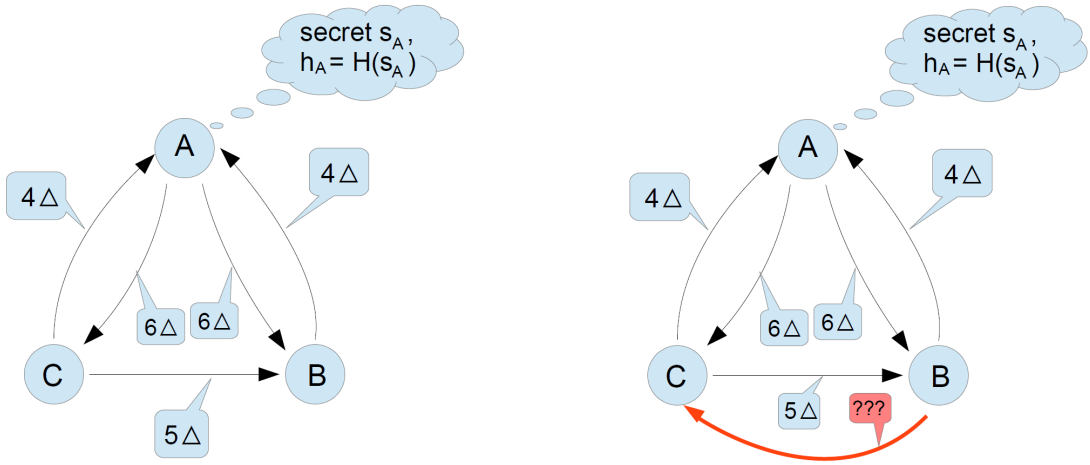

Figure 2.6: A is the leader, B and C followers. Timeouts can be assigned when the follower subdigraph is acyclic (left) but not when it is cyclic (right)

Unfortunately we have to assign timeouts across a cycle that guarantees a gap of at least Δ between entering and leaving arc, due to the fact that there is more than one leader this formular does not work in this case. This is because the subdigraph of any leader's follower has a cycle (see righthand side of figure 2.6). The general timed hashlock digraph can be replaced with a slight different version including leaders and followers to get a mechanism that allows us to assign different timeouts to different paths though. The modified version of the digraph that we need to include the leaders and followers model will be described next. Let $L = \{v_0, ..., v_n\}$ be the feedback vertex set for digraph $\mathcal{D}$, called the leaders. Each leader v_i creates a contract with secret s and a hashlock value $h_i = H(s_i)$, yielding a hashlock vector $(h_0, \ldots, h_n)$ with values assigned to each arc. A hashkey for h on arc(u, v) is a triple(s, ρ, σ) with s called the secret $h = H(s)$, ρ is a path $(u_0, \ldots, u_k)$ in digraph $\mathcal{D}$ where $u_0 = v$ and u_k is the leader who initially created the secret s, and $\sigma = sig(\cdots sig(s, u_k), \ldots, u_0)$ is the result, in case all parties have successfully signed s in their path. After the start of the protocol hashkey (s, ρ, σ) will time out at $(diam(\mathcal{D}) + |\rho|) \cdot \Delta$ and it will unlock h on (u, v) as long its available before it times out. Each arc triggers, when all needed hashlocks are unlocked. The hashlock itself times out on an arc, when all hashkeys on the current arc have timed out. Figure 2.7 illustrates the partial hashkeys for a two leader digraph.

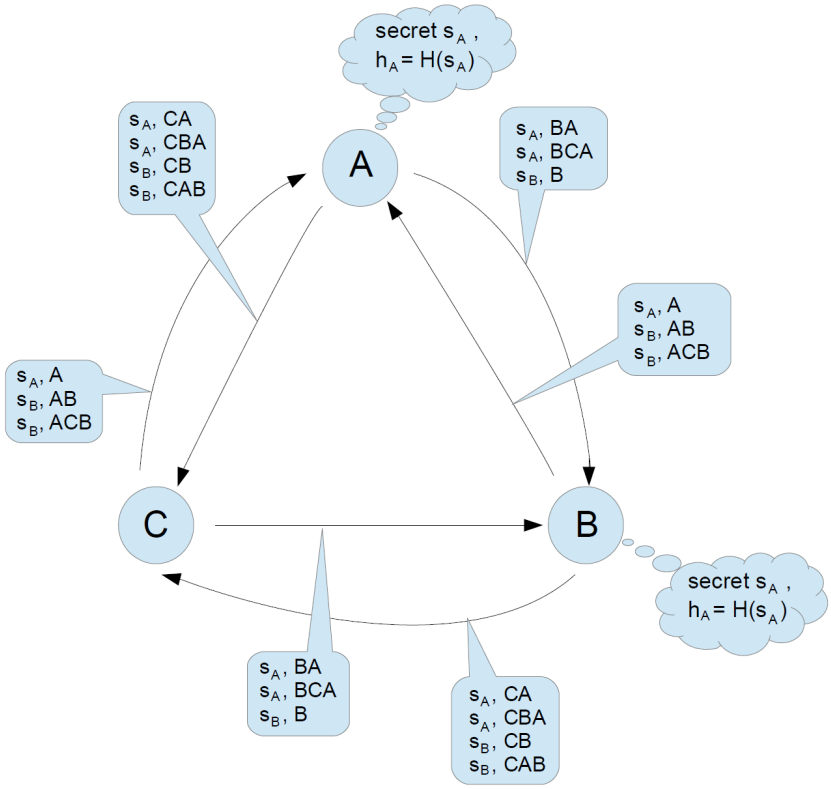

Figure 2.7: Hashkey paths for arcs of two-leader digraph

2.3.3 *The Protocol*

The protocol for atomic cross-chain swaps has two phases. In phase one the instances of the swap contract are propagated through the digraph $\mathcal{D}$, starting with the leaders. The implementation and its code are discussed later in chapter 5. As soon as a party observes a published contract the current party verifies the contract for correctness of the swap, if verification fails the party abandons the contract. Below the phase one protocol for leaders and followers is shown:

Protocol for the leaders:

1. Publish a contract on every arc that leaves the leader party, then
2. wait until contracts have been published and mined on all arcs entering the leader.

Protocol for the followers:

1. Wait until correct contracts are published and mined on all arcs entering the vertex, then
2. publish the contracts on all arcs leaving the vertex.

For an example refer to figure 2.7, which shows how contracts are propagated in a two leader swap digraph.

In phase two secrets are disseminated via hashkeys between the participating parties of the swap, while the contracts are propagated from party to counterparty in the direction of the arcs, hashkeys propagate in the opposite direction from counterparty to party. To acquire the assets held by the contracts, each party is motivated to trigger the contracts on entering arcs. In order to complete a swap and come to an understanding how the secret s_i generated by leader v_i is propagated we can trace it through the phase. At the start of phase two v_i calls unlock(s_i, v_i, $sig(s_i, v_i)$) on each contract of an entering arc. In this case the function's arguments are the hashkeys and v_i is a degenerate path. If any other party v observes that hashlock h_i has been unlocked by a call to unlock(s_i, ρ, σ) on a leaving arc's contract fro the first time, then is calls unlock(s_i, $v + \rho$, $sig(\sigma, v)$) at each entering arc's contract. The propagation of s_i is complete, when h_i has been unlocked on all arcs or if h_i has timed out. Herlihy has formally proven that the following statements apply to the atomic cross-chain swap protocol [19]:

- Assuming that all parties stick to the protocol, the time for triggering every contract from when the protocol started is $2 \cdot diam(\mathcal{D}) \cdot \Delta$.
- None of the conforming parties ends up UNDERWATER (loosing assets or ending up worse).
- The space complexity is $O(|A|^2)$, measured as the number of bits stored on all blockchains for $\mathcal{D} = (V, A)$ with leaders $L \subset V$.
- The set L of leaders is a feedback vertex set in $\mathcal{D}$ for any uniform swap protocol based on hashed timelocks.

3

RELATED WORK

WRITE ABOUT RELATED WORK HERE

theres also research specifically for ethereum private sidechains [32]

CROSS CHAIN SWAPS ARE WELL KNOWN CONCEPT IN BLOCKCHAIN COMMUNITY [4,6,9,20,21,27] -> explain further in related work tho!!

4

REQUIREMENT ANALYSIS AND CONCEPT

WRITE ABOUT REQUIREMENT ANALYSIS AND CONCEPT HERE

5 IMPLEMENTATION

WRITE ABOUT IMPLEMENTATION HERE

5.1 REFERENCE ARCHITECTURE

SHOW FIGURE FOR REFERENCE ARCHITECTURE HERE

5.2 SMART CONTRACTS

WRITE ABOUT SMART CONTRACT IMPLEMENTATIONS HERE

5.3 MIDDLEWARE (INTERMEDIATE PARTY)

WRITE ABOUT MIDDLEWARE HERE

5.4 PROOF OF CONCEPT (AKA TESTS?)

EXPLAIN TESTING AND PROOF OF CONCEPT HERE

6

FUTURE RESEARCH AND OUTLOOK

WRITE ABOUT FUTURE RESEARCH AND OUTLOOK HERE non interactive proofs of poW [21] sidechains [23] polkadot [39] chain interoperability [9] sharding [10] plasma [31] z-snarks [18] towards secure interoperability with smart contracts [16]

caught in chains mention open tasks from paper [6] pegged sidechains [3] sidechains and interoperability [20]

DeXTT: Deterministic Cross-Blockchain Token Transfers [7]

COSMOS [25] and tendermint [24] buchmann [8] PBFT [13]

pick some current research and make MB different chapters

Table 1. Cross-chain technology comparison Cross-chain technology Algorithmic principle Shortcoming Notary Transfer of assets to notary accounts Weak centralization Side chain Read the data in the side chain Verification difficulties Relay chain Establishing relay chain Data redundancy

Notary or multi-signature scheme is a weak centralized solution, transfer digital assets by locking assets in the main chain to multiple specific addresses and requiring signatures from multiple notaries. Ripple of Interledger protocol [2] enables assets of different blockchains to be transferred across chains through trusted third parties. Notary or multi-signature scheme is the simplest way to realize cross-chain asset transfer, but the security of asset transfer depends on the honesty of notaries. Side chain technology transfers digital assets by reading the data in the main chain to verify the authenticity of transaction payments. BTC Relay project is to realize crosschain payment by verifying Bitcoin SPV payment path to trigger the execution of ETH smart contract [3]. Relay technology is to link existing blockchain projects by building a relay chain to realize asset transfer of different blockchains. COSMOS [4] and Polkadot [5] projects are to build cross-chain platform of blockchain through the combination of notary and relay technology.

7

CONCLUSION

WRITE THE CONCLUSION HERE

8

TERMINOLOGY

EXPLAIN THE TERMINOLOGY HERE

Address Alt-Coin App Asset Atomic Transaction Autonomous Object Block Block Time Bytecode Cryptocurrency CryptoNote Contract External Actor Ether Ethereum Virtual Machine Gas Genesis Block Hash Padding - what is padding in sha-256 context Protocol - what is a protocol inctroducing ethereum and solidity paper Public and Private Key Public and Private Blockchain Storage State Solidity Transaction Token Nonce Miner

BIBLIOGRAPHY

[1] ISO/IEC 10118-3. *Information technology - security techniques - hash functions - Part 3: Dedicated hash-functions*. Tech. rep. 2003.

[2] Kurt M Alonso. *Zero to Monero*. 2020.

[3] 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." In: *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014).

[4] Adam Back et al. "Hashcash-a denial of service counter-measure." In: (2002).

[5] Jørgen Bang-Jensen and Gregory Gutin. "Theory, algorithms and applications." In: *Springer Monographs in Mathematics, Springer-Verlag London Ltd., London* 101 (2007).

[6] Michael Borkowski, Christoph Ritzer, Daniel McDonald, and Stefan Schulte. "Caught in chains: Claim-first transactions for cross-blockchain asset transfers." In: *Technische Universität Wien, Whitepaper* (2018).

[7] Michael Borkowski, Marten Sigwart, Philipp Frauenthaler, Taneli Hukkinen, and Stefan Schulte. "DeXTT: Deterministic Cross-Blockchain Token Transfers." In: *IEEE Access* 7 (2019), pp. 111030–111042.

[8] Ethan Buchman. "Tendermint: Byzantine fault tolerance in the age of blockchains." PhD thesis. 2016.

[9] Vitalik Buterin. "Chain interoperability." In: *R3 Research Paper* (2016).

[10] Vitalik Buterin. "On sharding blockchains." In: *Sharding FAQ* (2017).

[11] Vitalik Buterin et al. "Ethereum white paper." In: *GitHub repository* 1 (2013), pp. 22–23.

[12] Vitalik Buterin et al. "Ethereum: A next-generation smart contract and decentralized application platform." In: *URL https://github. com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper* (2014).

[13] Miguel Castro, Barbara Liskov, et al. "Practical Byzantine fault tolerance." In: *OSDI*. Vol. 99. 1999. 1999, pp. 173–186.

[14] Gary Chartrand, Linda Lesniak, and Ping Zhang. *Graphs & digraphs*. Vol. 39. CRC press, 2010.

[15] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. "Merkle-Damgård revisited: How to construct a hash function." In: *Annual International Cryptology Conference*. Springer. 2005, pp. 430–448.

[16] Gaby G Dagher, Chandra L Adhikari, and Tyler Enderson. "Towards Secure Interoperability between Heterogeneous Blockchains using Smart Contracts." In: *no. November* (2017).

[17] Chris Dannen. *Introducing Ethereum and Solidity*. Vol. 1. Springer, 2017.

[18] Alberto Garoffolo, Dmytro Kaidalov, and Roman Oliynykov. "Zendoo: a zk-SNARK Verifiable Cross-Chain Transfer Protocol Enabling Decoupled and Decentralized Sidechains." In: *arXiv preprint arXiv:2002.01847* (2020).

[19] Maurice Herlihy. "Atomic cross-chain swaps." In: *Proceedings of the 2018 ACM symposium on principles of distributed computing*. 2018, pp. 245–254.

[20] Sandra Johnson, Peter Robinson, and John Brainard. "Sidechains and interoperability." In: *arXiv preprint arXiv:1903.04077* (2019).

[21] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. "Non-Interactive Proofs of Proof-of-Work." In: *IACR Cryptology ePrint Archive* 2017.963 (2017), pp. 1–42.

[22] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. "Ouroboros: A provably secure proof-of-stake blockchain protocol." In: *Annual International Cryptology Conference*. Springer. 2017, pp. 357–388.

[23] Aggelos Kiayias and Dionysis Zindros. "Proof-of-work sidechains." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2019, pp. 21–34.

[24] Jae Kwon. "Tendermint: Consensus without mining." In: *Draft v. 0.6, fall* 1.11 (2014).

[25] Jae Kwon and Ethan Buchman. "A Network of Distributed Ledgers." In: *Cosmos, dated* (2018), pp. 1–41.

[26] Zujian Li and Zhihong Zhang. "Research and Implementation of Multi-chain Digital Wallet Based on Hash TimeLock." In: *International Conference on Blockchain and Trustworthy Systems*. Springer. 2019, pp. 175–182.

[27] Gregory Maxwell and Andrew Poelstra. "Borromean ring signatures." In: *Accessed: Jun* 8 (2015), p. 2019.

[28] Mark Miller. "The Future of Law." In: *paper delivered at the Extro* 3 (1997).

[29] Satoshi Nakamoto and A Bitcoin. "A peer-to-peer electronic cash system." In: *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* (2008).

[30] Shen Noether, Adam Mackenzie, et al. "Ring confidential transactions." In: *Ledger* 1 (2016), pp. 1–18.

[31] Joseph Poon and Vitalik Buterin. "Plasma: Scalable autonomous smart contracts." In: *White paper* (2017), pp. 1–47.

[32] Peter Robinson, David Hyland-Wood, Roberto Saltini, Sandra Johnson, and John Brainard. "Atomic crosschain transactions for ethereum private sidechains." In: *arXiv preprint arXiv:1904.12079* (2019).

[33] Nick Szabo. "Formalizing and securing relationships on public networks." In: *First Monday* (1997).

[34] Don Tapscott and Alex Tapscott. *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin, 2016.

[35] Nicolas Van Saberhagen. *CryptoNote v 2.0*. 2013.

[36] Henk CA Van Tilborg and Sushil Jajodia. *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2014.

[37] Dejan Vujičić, Dijana Jagodić, and Siniša Randić. "Blockchain technology, bitcoin, and Ethereum: A brief overview." In: *2018 17th international symposium infoteh-jahorina (infoteh)*. IEEE. 2018, pp. 1–6.

[38] Gerhard Weikum and Gottfried Vossen. *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*. Elsevier, 2001.

[39] Gavin Wood. "Polkadot: Vision for a heterogeneous multi-chain framework." In: *White Paper* (2016).

[40] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger." In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.