NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Cryptocurrencies and Mixing Services

Ioanna P. Giannikou

Supervisor (or supervisors): **Lazaros Merakos,** Professor

ATHENS

JUNE 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Cryptocurrencies and Mixing Services

Ιωάννα Π. Γιαννίκου

Επιβλέπων: **Λάζαρος Μεράκος,** Καθηγητής

ΑΘΗΝΑ

JUNE 2021

BSc THESIS

Cryptocurrencies and Mixing Services

Ioanna P. Giannikou

S.N.: 1115201600032

SUPERVISOR: **Lazaros Merakos,** Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Cryptocurrencies and Mixing Services

Ιωάννα Π. Γιαννίκου
Α.Μ.: 1115201600032

ΕΠΙΒΛΕΠΩΝ: **Λάζαρος Μεράκος,** Καθηγητής

ABSTRACT

This BSc thesis presents briefly the technical characteristics of some cryptocurrencies and the basic properties and mechanisms by which they operate and the support or not of smart contracts. The cryptocurrencies presented are the following: Bitcoin, Ethereum, Ripple (XRP), Dash, Litecoin, Namecoin. Moreover, this BSc thesis focuses on cryptography techniques and mixing services. A cryptocurrency mixing service is a service offered for mixing potentially identifiable or "infected" encryption funds with others, in order to overshadow the trail back to the original source of the fund. There are various protocols used to achieve sender anonymity, each with its own mode of operation, pros, cons, and capabilities. The protocols that are analyzed and compared are the following: Mixcoin, Blindcoin, Tumblebit, Coinjoin, Coinshuffle, Xim, Mobius and Mixeth.

SUBJECT AREA: Mixing Services

KEYWORDS: cryptocurrencies, mining, consensus mechanisms, cryptographic techniques, mixing strategies, mixing protocols, smart contracts.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία παρουσιάζει περιληπτικά τα τεχνικά χαρακτηριστικά ορισμένων κρυπτονομισμάτων, τις βασικές ιδιότητες και μηχανισμούς με τους οποίους λειτουργούν και την υποστήριξη ή όχι από smart contracts. Τα κρυπτονομίσματα που παρουσιάζονται είναι τα εξής: Bitcoin, Ethereum, Ripple (XRP), Dash, Litecoin, Namecoin. Επιπρόσθετα, η εργασία επικεντρώνεται στις τεχνικές κρυπτογραφίας και στις υπηρεσίες ανάμειξης. Η υπηρεσία ανάμειξης κρυπτονομισμάτων είναι μια υπηρεσία που προσφέρεται για τη μίξη δυνητικά αναγνωρίσιμων ή «μολυσμένων» κεφαλαίων κρυπτογράφησης με άλλους, έτσι ώστε να επισκιάσει το ίχνος πίσω στην αρχική πηγή του ταμείου. Υπάρχουν διάφορα πρωτόκολλα που χρησιμοποιούνται για να πετύχουν την ανωνυμία του αποστολέα, καθένα από αυτά με τον δικό του τρόπο λειτουργίας, πλεονεκτήματα, μειονεκτήματα και δυνατότητες. Τα πρωτόκολλα που αναλύονται και συγκρίνονται μεταξύ τους είναι τα εξής: Mixcoin, Blindcoin, Tumblebit, Coinjoin, Coinshuffle, Xim, Mobius και Mixeth.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Mixing Services

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: cryptocurrencies, mining, consensus mechanisms, cryptographic techniques, mixing strategies, mixing protocols, smart contracts.

This work is dedicated to my father.

ΕΥΧΑΡΙΣΤΙΕΣ

I would like to thank Dr. Dionysis Xenakis for his supervision and valuable contribution to the completion of this thesis.

CONTENTS

LIST OF IMAGES

LIST OF TABLES

PREFACE

1. INTRODUCTION

Money is a medium dating back at least to Asia Minor in the 7th Century B.C.E. in the form of single-sided electrum coins issued by the Lydian Empire, and possibly earlier to clay tokens used by the Sumerian Empire in 8,000 B.C.E.2. If we expand the definition to include barter, we can date money back to 12,000 B.C.E. when obsidian and cattle served as media of exchange [1]. One way to conceive of money is as an IOU (“I owe you”) that is denominated in a nation’s unit of account and deemed acceptable by somebody other than its issuer. In other words, it has the particularity that it is generally accepted by all in the economy in exchange for goods and services. Money is essential to the functioning of the modern economy, but its nature has changed significantly over time. Our modern economy relies heavily on digital means of payments. While technology has given us more control over our financial lives (online shopping, online banking, mobile payments, automatic deposits), transactions had to keep up with the new data and all known conventional currencies were no longer able to meet the increased amount of new trading needs. For this reason, cryptocurrencies emerged as a digital alternative to more traditional methods of exchange like cash or credit cards. More and more companies are accepting the payments through cryptocurrency these days. The best part of using cryptocurrency is that every transaction you carry out with the recipient would be easy, secure and confidential with low transaction fee.

Establishing a definition of cryptocurrencies is no easy task. Cryptocurrencies has become a “buzzword” to refer to a wide array of technological developments that utilize a technique better known as cryptography. In simple terms, cryptography is the technique of protecting information by transforming it (i.e. encrypting it) into an unreadable format that can only be deciphered (or decrypted) by someone who possesses a secret key [2]. Cryptocurrencies use peer-to-peer (P2P) networks to disseminate system information while keeping the whole system as much decentralized as possible. This means that there is not only no need for any previous relationship or trust between involved persons, but also there is no need for a central trust authority as regulator. This mechanism provides the creation of valid transactions with high resilience and security.

The idea of having a digital currency is not a new one. Many attempts at creating a digital currency had taken place before the creation of cryptocurrencies, but most of them had to deal with the problem of double spending. A digital asset should be used only once in order to prevent copying it and effectively counterfeiting it. The birth of Bitcoin was the first implementation of the idea of a decentralized, reliable digital currency. Bitcoin was released as open-source software in 2009 and since then the cryptocurrency market currently hosts almost 7000 digital currencies and is still expanding. Some of them were established in the cryptocurrency market such as Ethereum, Ripple, Litecoin and Dash, while others are not used that much. Although, there are differences in the implementation and design between cryptocurrencies, all of them have a design principle based on cryptography and utilize forms of blockchain technology to gain decentralization and protection against attacks. This decentralized nature of blockchain technology protects cryptocurrencies from control by any central authority. Another similarity between cryptocurrencies is the fact that they are all based on a P2P computer network, which hosts transactions of any kind and the amount of coins to be minted stay constant, despite their variable value.

1.1 Basic cryptocurrency terminology

1.1.1 Blockchain

A blockchain is a type of database, which stores information in blocks that are linked using cryptography. These blocks are distributed to many people instead of being run by one government, organization, group or person. Every block is used to prove that a group of people has reached an agreement about something. The blockchain is being simultaneously created and maintained by thousands of individuals all at the same time.

When a new block is added to the network, each network user gets a new copy of it. The fact that every single person has and maintains his own copy of the blockchain is known as “decentralized”. Decentralized blockchains are immutable, which means that the data entered is irreversible. A block can store different types of information, but its use has been established for transactions and their characteristics. In a blockchain system, all the data traded, include currency and information of the traders, have a digital form.

In other words, a blockchain is a distributed consensus system that allows transactions through cryptography, computing, and the network users themselves. These transactions are quickly authenticated, permanent, very secure, preventing manipulation. The main reasons why these transactions are secure and fast are the technology that a blockchain uses and the number of people maintaining it. Blocks are created one after the other and chained together in sequence. This chain of blocks has a chronological order and each block contains a timestamp. The network relies on the history of the past transactions to certify the validity of a new one, using a hash function for each block.

An example in how Blockchain works step by step, as described in [77]:

1. A and B conducts a transaction.
2. Cryptographic keys are assigned to the transaction that both A and B holds.
3. A new block is created representing the transaction.
4. Transaction is broadcasted and validated by a distributed network.
5. The block is then appended to the blockchain, creating a permanent record of the transaction.
6. Transaction between A and B is completed.

1.1.2 Nodes

A blockchain consists of blocks of data that are stored on nodes. A blockchain theoretically exists on nodes, as they ensure its operation and survival. A node can be any kind of device such as computers, laptops, phones. All nodes are interconnected. Nodes store, spread, preserve the blockchain data and constantly exchange the latest data with each other, so that all nodes remain up to date. The work of accepting transactions as valid and adding them to the blockchain is done by nodes. In other words, nodes validate and forward the transaction to their own peers until it reaches all network nodes.

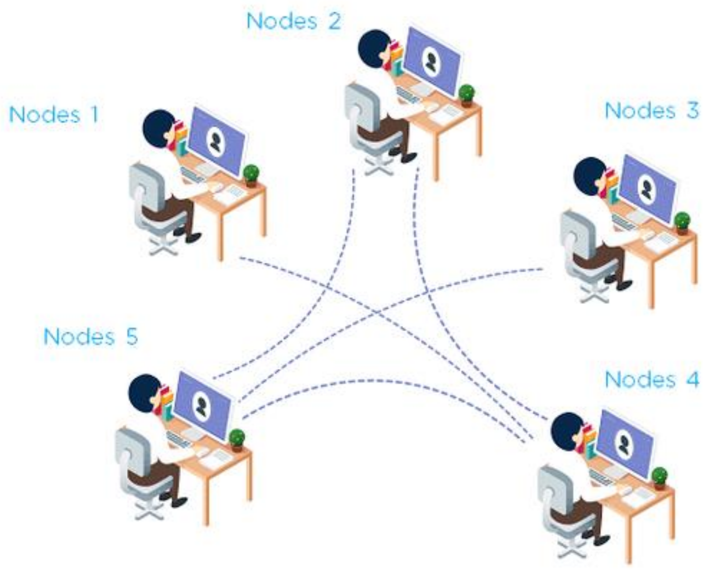


Image 1: Nodes [78]

1.1.3 Mining

Mining, also known as minting, is defined as the computer process of validating transactions, creating a new block of transactions into the blockchain, and it is a mechanism by which new currencies enter the network. Some specific nodes, known as miners, spend huge amounts of computing power and energy for this process. A financial reward is been given to miners for their work, and this reward decreases transaction fees. For this reason, they are motivated to give priority to the validation of transactions that include a higher fee and remain trustworthy.

Every cryptocurrency that implements mining does not follow the same mining process, and some of them have not that process at all. The current hash rate only for the case of Bitcoin mining amounts to 86.002.554 TH / s. Hash rate is the way we measure how much computing power everyone around the world is contributing toward mining Bitcoin.

At first, the Bitcoin mining process was done exclusively by CPUs (Central Processing Unit), which are integrated in computers and other devices. Later, CPUs gave their place to GPUs (Graphics Processing Unit) because data access was faster due to their structure and the ability of parallel execution. Other technologies used for this process are FPGAs (Field-Programmable Gate Array), which have better hash rate and consume less energy and ASICs (Application-Specific Integrated Circuit) that are specially designed for the mining process and produce the maximum possible number of hashes by running a hash function in parallel. Nowadays, the only way to implement a successful mining is by using ASICs. Solo mining is when a miner completely does his task of mining operations on his own without joining a pool. All these blocks are created in such a way that the work is completed with the miner’s credit, which means that it might be unprofitable for individual users. Pool mining was appeared to deal with this problem. In pool mining, a set of miners contribute their computing power to the pool, generating the solution to a given block. Each user, who joins the pool, is obliged to pay a small participation fee to his administrator. After the mining process, they share the financial reward, which is distributed to each user depending on the hash rate contribution to the pool.

1.1.4 Wallet

A blockchain wallet is a digital wallet that allows users to store and transfer their cryptocurrencies and specifically bitcoin and ether. It charges dynamic fees, meaning that transaction fees may differ because of factors such as the transaction size.

1.1.5 Hashing

A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The input data become a secret, as the output information is unreadable. By using hash functions, storing and finding information is much more quickly, as hashes are shorter and easier to find.

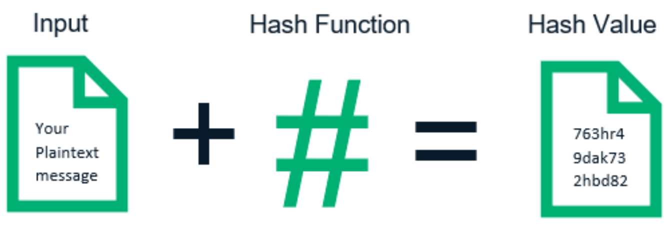


Image 2: Hashing [79]

1.1.6 Forks

Blockchain forks are essentially a split in the blockchain network. In case of blockchain, forks can occur when the mining process diverges simultaneously the blockchain into two potential paths (two versions of the blockchain). It is up to the PoW systems to give solution to these temporary forks. They decide which is the most appropriate path, comparing the timestamp or the path’s length.

Forks based on the software can be distinguished in soft and hard forks. Both lead to permanent changes to the protocol rules of a cryptocurrency. A difference between them is that in case of hard fork upgrading the protocol software leads the new version to be not compatible with the older one. Moreover, in hard forks all network nodes are required to upgrade their current protocol in order to continue participating in the network and blocks that have been verified by nodes that have not been upgraded are considered invalid. This requirement for a total upgrade does not exist in the case of soft forks, as the blocks can follow both new and old rules but if the new rules are violated by a block, this block may end up as “stale”.

1.1.7 Merkle Tree

In many cases of cryptocurrencies, Merkle trees (or similar hash trees) are used. A Merkle Tree is a tree in which every leaf node (node without descendants) corresponds to the hash value of a data block and every non-leaf node (node with descendants) is labelled with the hash value of its corresponding child nodes (concatenates the children’s hash values).

This data structure allows efficient and secure verification of the contents of large data structures. Also, it uses small amounts of memory and provide an efficient way to certify data validity and integrity.

The only way to certify the content of many blocks is by storing the hash value of merkle root in the blockchain’s block header. Network users can easily decide if a block matches or not by comparing the hash values of the tree root.

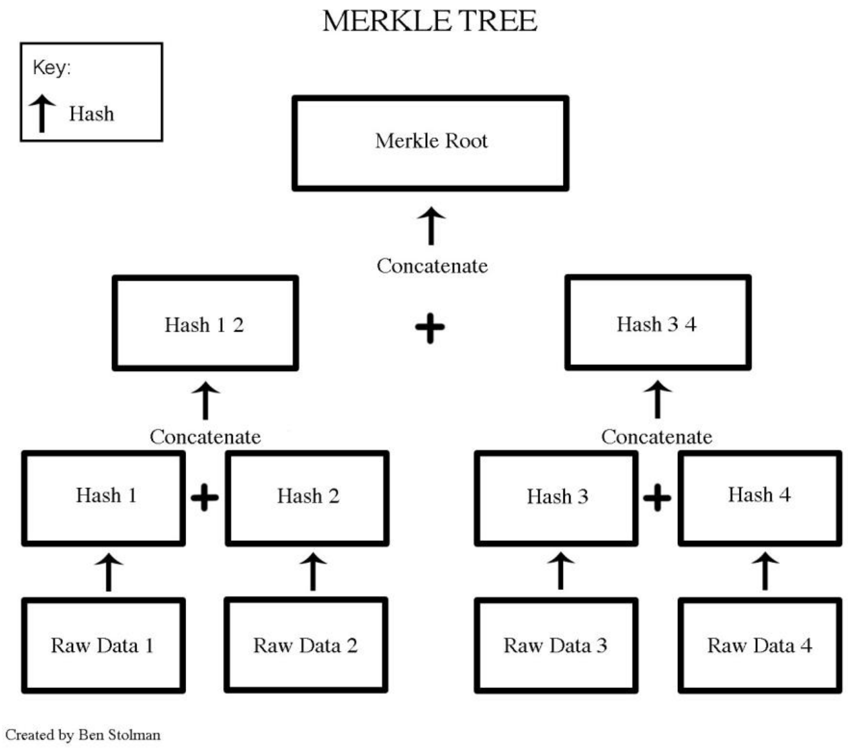


Image 3: Merkle Tree [80]

1.1.8 Directed Acyclic Graph

A DAG (Directed Acyclic Graph) is a directed graph with no directed cycles. This means that it is impossible to traverse the entire graph starting at one edge, because edges of the directed graph go only one way. That is, it consists of vertices and edges, with each edge directed from one vertex to another, such that there is no way to start at any vertex v and follow a consistently-directed sequence of edges that eventually loops back to v again. Equivalently, a DAG is a directed graph that has a topological ordering, a sequence of the vertices such that every edge is directed from earlier to later in the sequence [3], [4].

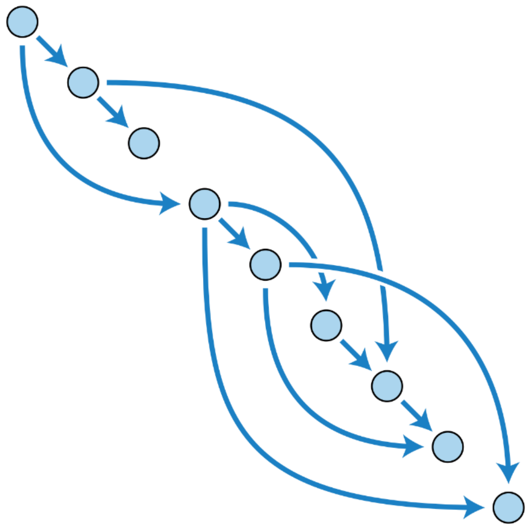


Image 4: Directed Acyclic Graph [81]

1.1.9 51 percent attack

According to [76], a 51% attack refers to an attack on a blockchain (most commonly bitcoins, for which such an attack is still hypothetical) by a group of miners controlling more than 50% of the network's mining hash rate or computing power.

The attackers would be able to prevent new transactions from gaining confirmations, allowing them to halt payments between some or all users. They would also be able to reverse transactions that were completed while they were in control of the network, meaning they could double-spend coins.

They would almost certainly not be able to create new coins or alter old blocks. A 51% attack would probably not destroy bitcoin or another blockchain-based currency outright, even if it proved highly damaging.

1.2 Consensus assurance mechanisms

1.2.1 Proof of Work (PoW)

Proof of Work (PoW) is a form of cryptographic zero-knowledge protocol, which proves that a certain amount of computational effort has been expended for some purpose. At first place, it used to deal with spam in email services. The most famous application of PoW is Bitcoin and since then it has been used in other bitcoin-based cryptocurrencies, such as Litecoin and Ethereum.

PoW is used in a blockchain network as the original consensus algorithm, which confirms transactions and produce new blocks to the chain. Using this technique, miners compete against each other to complete transactions on the network and get rewarded for their hard work. In order to succeed this, they are called to solve a complex mathematical puzzle. This task requires continuous calculations and enough computing power to provide an answer, which is called hash. The complexity of that puzzle depends on network load, power and the number of users. The hash of the previous block is contained by each hash of the current block. This increases the security and makes it difficult for any blocks to be violated. A new block is formed, if a miner manages to give a solution to the puzzle, and then the transactions are placed in this block and considered as confirmed.

1.2.1.1 Proof of Work Advantages

PoW advantages:

- PoW prevents the network from Sybil and DoS Attacks, as these attacks need waste of computational power and time and money to be executed.
- PoW systems emphasize on having large computational power in order to solve the puzzles and form new blocks. This means that it does not matter how much money there is in someone's wallet for making decisions.

1.2.1.2 Proof of Work Disadvantages

Although, PoW seems to have some important benefits, there are some drawbacks too.

PoW disadvantages:

- Mining process requires an expensive equipment and needs large amounts of power, that only specific users can afford, which means that the system is threatened to be centralized.
- In a case when a user or a group of users controls most of the mining power, a 51 percent attack can take place.
- PoW systems spend a lot of energy to find the puzzle’s solutions, but these solutions are calculated to be calculated. In other words, they have no application in any field.

1.2.2 Proof of Stake (PoS)

Proof of Stake (PoS) was created as an alternative to Proof of Work (PoW). This consensus mechanism is based on a certain number of cryptocurrencies (stake), that each node (now called validator) chooses to bind to the network as a deposit. In other words, PoS algorithms achieve consensus by requiring users (validators) to stake an amount of their tokens. Validators are responsible for the creation of a new block and by using this consensus mechanism they can decide which block of transactions is valid for the blockchain and get rewarded for doing so. The strength of their vote depends on the stake, that they contributed. Moreover, the higher their initial contribution is, the greater the chance of getting rewarded. If a validator does not follow the consensus rules, then the amount set as a stake is withheld.

1.2.2.1 Proof of Stake Advantages

PoS advantages:

- PoS algorithms are energy efficient and a greener option.
- PoS has low barrier to entry. No expensive investment in hardware is required.
- PoS cryptocurrencies are generally faster than PoW cryptocurrencies.

1.2.2.2 Proof of Stake Disadvantages

Although, PoS seems to have some important benefits, there are some drawbacks too. PoS disadvantages:

- PoS is unproven in terms of long-term sustainability.
- In PoS, users who hold a large amount of coins can have an outsized influence on the consensus process.

1.2.2.3 The “Nothing on Stake” Problem

Although PoS seems to be the best option for a cryptocurrency consensus algorithm, there is an obstacle that discourages the use of this consensus mechanism as an ideal. This problem is known as “Nothing on Stake”. The main idea of this problem is that in a case of a blockchain split by fork, block generators have nothing to lose by supporting different blockchains.

1.2.3 PoW vs PoS

In the following table, there is a comparison between PoW and PoS.

Table 1: PoW vs PoS

PoW	PoS
PoW algorithms need huge amounts of energy for the mining process.	PoS algorithms are energy efficient and a greener option.
In a PoW system, attackers don’t lose their hardware when attempting 51% attacks.	In a PoW system, attackers must offer their stake in order to attempt a 51% attack.
PoW has an exponential increase in reward per investment, leading to a very real threat of centralization.	PoS has a linear increase in reward per investment, so the decentralization is not in danger.
There is not “Nothing on Stake” problem, because users would have to split their computational resources in order to support both sides of the fork.	There is “Nothing on Stake” problem, because validators’s stakes would be duplicated onto both chains, meaning they could potentially claim twice the amount of rewards.

Image 5: PoW vs PoS [82]

1.2.4 PoW/PoS Hybrid

It is already proved that mixing of two or more existing protocols (hybrid protocol) can make the network enough resistive to attacks. Following this idea, hybrid PoW and PoS took place. The objective of hybrid PoW and PoS systems is to use the benefits of each mechanism and use them to improve each other’s weaknesses. Not only does this design not require high energy consumption, but also increases the costs of attacking the network because an attacker must deal with two distinct systems.

1.3 Description of basic cryptocurrencies

1.3.1 Bitcoin

Although the idea of a decentralized electronic currency existed before, bitcoin was the first-ever cryptocurrency to come into actual use. Bitcoin is a decentralized cryptocurrency originally described on 31 October 2008 in a paper titled “Bitcoin: A Peer-to-Peer Electronic Cash System” by the presumed pseudonymous Satoshi Nakamoto. Later, on 3 January 2009, the bitcoin network came into existence with Satoshi Nakamoto mining the genesis block of bitcoin (block number 0), which had a reward of 50 bitcoins. Bitcoin was created, according to Nakamoto’s own words, to allow “online payments to be sent directly from one party to another without going through a financial institution.”

Bitcoin price today is $33,538.68. The current CoinMarketCap ranking is #1, with a live market cap of $628,702,832,742. The total number of coins in circulation is 18,745,606BTC coins.

Each new currency that is created should follow a set of rules, otherwise it is rejected and acquires no value at all. New coins are put into circulation by the mining process. The validation of a block of transactions and its integration to the blockchain involves the influx of new currencies into the network. As described in [27], the rate of block creation is adjusted every 2016 blocks to aim for a constant two-week adjustment period (equivalent to 6 per hour.) The number of bitcoins generated per block is set to decrease geometrically, with a 50% reduction every 210,000 blocks, or approximately four years. The result is that the number of bitcoins in existence will not exceed slightly less than 21 million. All these are arbitrary conventions and no explanation has ever been given by Satoshi Nakamoto, but there is an estimation that the total number of Satoshis (the smallest unit of the bitcoin currency) that can be mined is close to the maximum capacity of a 64-bit floating point number.

A miner is rewarded for the creation and union of new valid transaction blocks in the blockchain and by the fees that accompany the transactions. If all the coins are mined, the miners will continue their activity, motivated by the transaction fees.

Bitcoin owners have digital wallets, protected by cryptography and accessible only via private key. In a case of losing that key, that wallet may be permanently inaccessible, along with its contents, so there is a loss of coins that cannot be replaced.

Bitcoin has two type of transactions:

- **Coinbase transaction**: a unique type of bitcoin transaction that can only be created by a miner. It has no inputs, as new BTC coins enter the system and each new block includes one, as the first transaction that is mined on the network. It is also related to the reward of the successful miner.
- **Regular transaction**: a transaction for transferring existing BTC coins between network users.

A Transaction has the following fields:

- **nVersion**: the current transaction version (with value 1).
- **#vin**: the number of input data
- **hash/ transaction ID**: It is calculated from the dual application of SHA256 to the contents of the transaction and in the case of a regular transaction (RT) it is the hash value of the previous transaction.
- **n**: index of the transaction output indicated by the hash field
- **scriptSigLen (RT)**: the size of the scriptSig (RT) /

  coinbaseLen (CBT): the size of the coinbase (CBT) field
- **scriptSig (RT)**: the script that satisfies the transaction satisfaction condition /

  coinbase (CBT): encoding block height and other data
- **nSequence**: the serial number of the transaction entry
- **#vout**: the number of output data
- **nValue**: amount expressed in satoshis
- **scriptPubkeyLen**: the size of scriptPubKey
- **scriptPubkey**: the script that specifies the conditions under which the output of a transaction can be claimed
- **nLockTime**: timestamp of the "lock" of the transaction

A transaction is called final if one or both factors are true:

- The nLockTime field of a transaction is set to locked or the corresponding time period has elapsed.
- All transaction input is final.

Each block consists of a header and payload.

Header fields are:

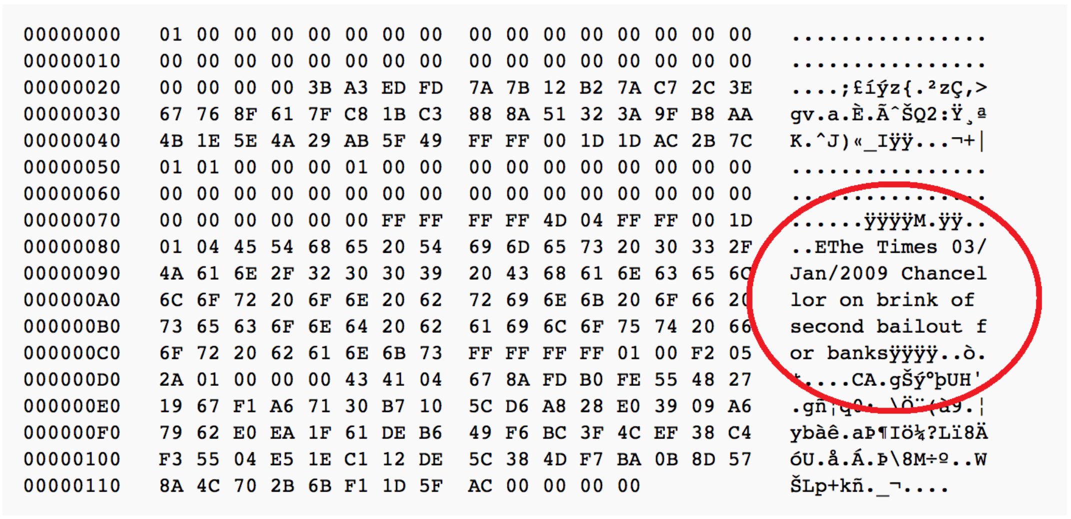
- **nVersion**: the current version of the block (with value 2). Each block with a different nVersion can neither be mined nor transmitted.
- **HashPrevBlock**: the hash value of the block header. Specifically, the entire header of the immediately preceding block is merged and entered as input to a dual SHA256 function.
- **HashMerkleRoot**: the hash value of the Merkle tree root used for blockchain transactions.
- **nTime**: the timestamp of creating the block.
- **nBits**: the target value for the proof of work.
- **nNonce**: the nonce factor for mining.

Payload fields are:

- **#vtx**: the number of transactions
- **Vtx**: the block of transactions

During the mining process, there is a possibility that two miners will succeed and create two valid blocks at the same time. This leads to a breakdown of the blockchain (fork), so the nodes must decide which one should be added to the blockchain, comparing each block's timestamp. The block, which has not been accepted within the bitcoin blockchain, is called stale or orphan. Its transactions are valid, and they are returned to the transaction pool on hold.

A genesis block is the first block of a blockchain, and it is a special case in that it does not reference a previous block. Modern versions of Bitcoin number it as block 0, though very early versions counted it as block 1.



```
00000000   01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000010   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000020   00 00 00 00 3B A3 ED FD  7A 7B 12 B2 7A C7 2C 3E   ....;£íýz{.²zÇ,>
00000030   67 76 8F 61 7F C8 1B C3  88 8A 51 32 3A 9F B8 AA   gv.a.È.Ã^ŠQ2:Ÿ¸ª
00000040   4B 1E 5E 4A 29 AB 5F 49  FF FF 00 1D 1D AC 2B 7C   K.^J)«_Iÿÿ...¬+|
00000050   01 01 00 00 00 01 00 00  00 00 00 00 00 00 00 00   ................
00000060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000070   00 00 00 00 00 00 FF FF  FF FF 4D 04 FF FF 00 1D   ......ÿÿÿÿM.ÿÿ..
00000080   01 04 45 54 68 65 20 54  69 6D 65 73 20 30 33 2F   ..EThe Times 03/
00000090   4A 61 6E 2F 32 30 30 39  20 43 68 61 6E 63 65 6C   Jan/2009 Chancel
000000A0   6C 6F 72 20 6F 6E 20 62  72 69 6E 6B 20 6F 66 20   lor on brink of 
000000B0   73 65 63 6F 6E 64 20 62  61 69 6C 6F 75 74 20 66   second bailout f
000000C0   6F 72 20 62 61 6E 6B 73  FF FF FF FF 01 00 F2 05   or banksÿÿÿÿ..ò.
000000D0   2A 01 00 00 00 43 41 04  67 8A FD B0 FE 55 48 27   *....CA.gŠý°þUH'
000000E0   19 67 F1 A6 71 30 B7 10  5C D6 A8 28 E0 39 09 A6   .gñ¦q0·.\Ö¨(à9.¦
000000F0   79 62 E0 EA 1F 61 DE B6  49 F6 BC 3F 4C EF 38 C4   ybàê.aÞ¶Iö¼?Lï8Ä
00000100   F3 55 04 E5 1E C1 12 DE  5C 38 4D F7 BA 0B 8D 57   óU.å.Á.Þ\8M÷º..W
00000110   8A 4C 70 2B 6B F1 1D 5F  AC 00 00 00 00            ŠLp+kñ._¬....
```

Image 6: Genesis Block [83]

The coinbase parameter (seen above in hex) contains, along with the normal data, the following text: “The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.”

This was probably intended as proof that the block was created on or after January 3, 2009, as well as a comment on the instability caused by fractional-reserve banking [31].

Bitcoin blockchain consists of serialized blocks, which contain permanently recorded information related to transactions. The last block is called block head. The mining process leads to the addition of new blocks to the blockchain and is repeated every 10 minutes or so. Miners follow the PoW algorithm. The decisions in the Bitcoin blockchain are made democratically. The longer blockchain is always considered more valid, because it takes a lot of computing power to create the longest version of blockchain and, it is supposed to be supported by the majority.

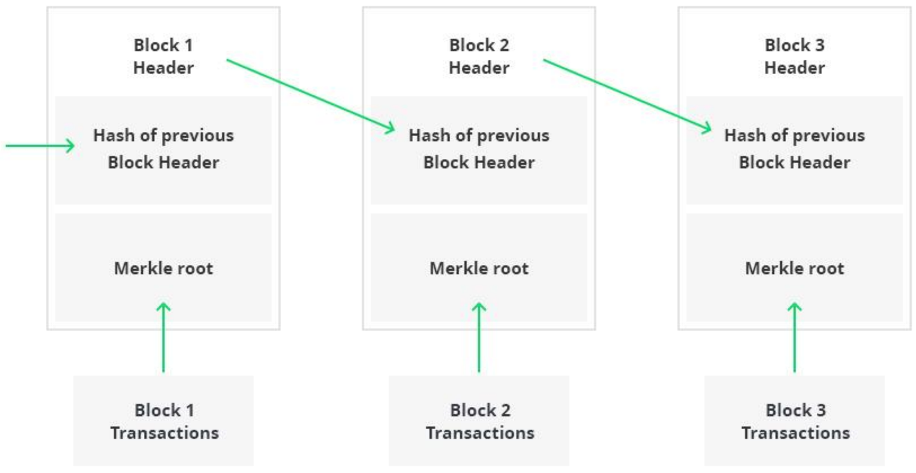


Image 7: Bitcoin Blockchain [84]

The bitcoin network is a peer-to-peer payment network that operates on a cryptographic protocol. Each node participates equally in the network and interacts with the others without using "special" nodes. Its network is decentralized, as it does not have a central authority to define or direct it. Every user has access to it, and it does not require new users to comply with a set of rules and restrictions. Moreover, the more users provide services on the network, the more resistance there is to malicious actions.

1.3.2 Ethereum

Ethereum is a decentralized open source blockchain with smart contract functionality. Ether (ETH) is the native cryptocurrency token of the Ethereum platform. It was proposed in late 2013 by Vitalik Buterin, a cryptocurrency researcher and programmer, but a few researchers have contributed to the development of the Ethereum platform, such as Dr. Gavin Good and Dr. Joseph Lubin. The development of the Ethereum network has gone through several distinct stages, and each stage marked the adoption of a series of important changes. As described in [32], these stages are:

1. **Frontier**: The first stage for the development of the Ethereum project was aimed mainly at people with a technical background due to the technical nature of the interaction interface of the network. Frontier is essentially the Ethereum platform in its most minimalistic form where people can purchase ether to upload their own software.

2. **Homestead**: In the second stage of its development, which was launched on March 14, 2016, Ethereum went a little further by introducing many protocol improvements to

speed up transactions on the networks. This stage involved the activation of more blocks, while laying the foundation for future upgrades.

3. **Metropolis**: The Metropolis stage will see changes to the network including the introduction of security updates and an increased degree of automation in smart contracts. This phase will also make the platform easier for the layperson by introducing programs that do not require the downloading of the entire blockchain locally.

4. **Ice Age or Serenity**: The Serenity phase envisages changing from a Proof of Work method to a Proof of Stake method in a bid to reduce energy consumption. The network will also be faster, more efficient and easier for beginners to understand.

Ethereum is the most actively used blockchain in the world. Ethereum price today is $593.21 USD. It is the second-largest cryptocurrency by market capitalization, with a market cap of $67,469,063,964 USD. It has a circulating supply of 113,736,136 ETH coins. In theory, it is possible for infinite coins to circulate if the system works perfectly, so max. supply is not available. About 60 million ethers have emerged from presale and about 12 million ethers were raised by the Ethereum foundation, in order to financially support those who contributed to the project and the equipment. The rest of them have emerged from the mining process as a reward to miners or reward for miners who found the correct answer, but their block did not be added to the blockchain (uncle/aunt reward).

The reward changes of each block did not result from any internal regulation of the protocol, as in the case of Bitcoin. Instead, they were proposed for a few reasons and eventually adopted. The rate of issuance of new coins also affected the speed of production of blocks.

As reported in [35], an Ethereum transaction refers to an action initiated by an externally-owned account.

A submitted transaction includes the following information:

- **Recipient**: the receiving address (if an externally-owned account, the transaction will transfer value. If a contract account, the transaction will execute the contract code).
- **Signature**: the identifier of the sender. This is generated when the sender's private key signs the transaction and confirms the sender has authorized this transaction.
- **Value**: amount of ETH to transfer from sender to recipient (in WEI, a denomination of ETH).
- **Data**: optional field to include arbitrary data.
- **gasLimit**: the maximum amount of gas units that can be consumed by the transaction. Units of gas represent computational steps.
- **gasPrice**: the fee the sender pays per unit of gas.

The first 3 factors are common to every cryptocurrency. The data field is utilized in cases where we have a smart contract implementation. Ethereum’s transactions utilize bandwidth, storage and computing power based on existing requirements, so the implementation of transaction fees is more complicated than bitcoin, thus protecting the system from DoS attacks.

Ethereum blocks consist of block header and transaction list. Block header has the following fields:

- **Parent hash**: is the hash value of the parent block header.
- **Ommers / Uncle hash**: is the hash value of the list of ommer / uncle blocks.
- **Beneficiary**: corresponds to a 20-byte address, to which all block rewards are transferred.
- **State Root**: is the hash value of the root node of the state trie, which are calculated from the moment that the transactions of a block are finished.
- **Transactions Root**: is the hash value of the root node of the trie structure, which includes every transaction in the block transaction list.
- **Receipts Root**: is the hash value of the root node of the trie structure, which contains the evidence for each transaction of the transaction list of the block.
- **Difficulty**: is the difficulty of each block. This amount is calculated from the difficulty of the previous block and its own timestamp.
- **Bloom Logs**: corresponds to a bloom filter.
- **Number**: corresponds to the number of ancestor blocks that exist behind the current block.
- **Gas Limit**: corresponds to the current maximum required cost in gas for the block.
- **Gas Used**: corresponds to the total amount of gas used in the block transactions.
- **Timestamp**: corresponds to the time the current block was created in Unix time.
- **Extra data**: is a byte-table, up to 32 bytes in size, which contains additional information about the block.
- **Mix Hash**: is a hash value, size 32 bytes, which certifies that a sufficient amount of computing effort has been spent within the specific block.
- **Ommer / Uncle Block Header**: corresponds to the header of uncle blocks.

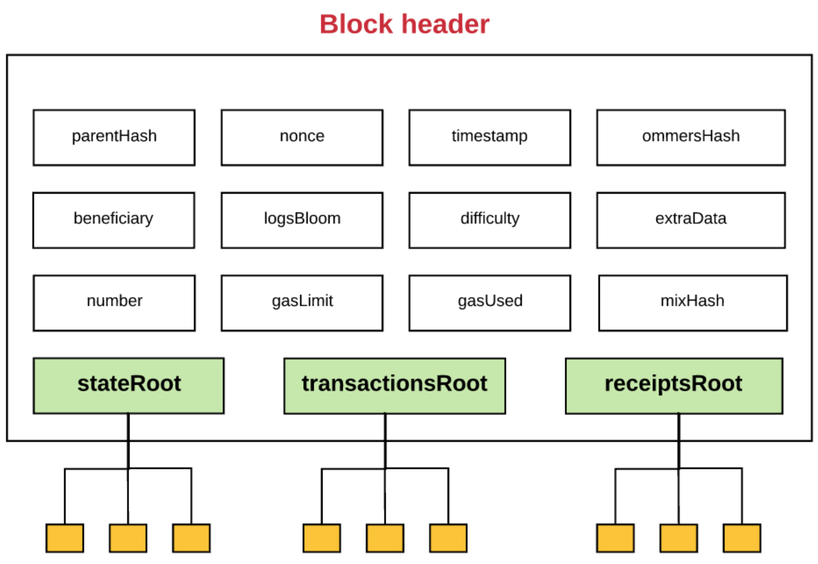


Image 8: Ethereum-Block Header [85]

For all hash values, the Ethereum platform utilizes the Keccak-256 cryptographic hash function.

An Ethereum account is an entity with an ether (ETH) balance that can send transactions on Ethereum. Accounts can be user-controlled or deployed as smart contracts [67].

Transactions are cryptographically signed instructions from accounts. An account will initiate a transaction to update the state of the Ethereum network. The simplest transaction is transferring ETH from one account to another [68].

The EVM’s (Ethereum Virtual Machine) physical instantiation can’t be described in the same way that one might point to a cloud or an ocean wave, but it does exist as one single entity maintained by thousands of connected computers running an Ethereum client. The Ethereum protocol itself exists solely for the purpose of keeping the continuous, uninterrupted, and immutable operation of this special state machine; It's the environment in which all Ethereum accounts and smart contracts live. At any given block in the chain, Ethereum has one and only one 'canonical' state, and the EVM is what defines the rules for computing a new valid state from block to block [66].

Ethereum blockchain essentially works as a transaction-based state machine. Ethereum also includes a copy of the list of transactions and the latest status.

A block of transactions is created and added to the Ethereum blockchain by the mining process. Ethereum uses a proof-of-work (PoW) consensus mechanism called Ethash, which utilizes a DAG for its implementation. The Ethereum protocol intends to use a PoS consensus algorithm (named Casper).

1.3.3 Ripple (XRP)

Ripple is the name of the company that provides money shipping services worldwide, and XRP is the digital currency adopted in its transactions. However, it is common to refer to the company (Ripple), the platform (RippleNet) and the currency (XRP) with the same name, Ripple. One big problem with traditional money is that it takes days to move value from network to network, and this process costs a lot. Using Ripple, banks can meet payment services of any size faster and with low transaction costs.

Ripple was conceived by Jed McCaleb and built by Arthur Britto and David Schwartz who then approached Ryan Fugger who had debuted in 2005 as a financial service to provide secure payment options to members of an online community via a global network [37].

The XRP Ledger is an online system for payments, powered by a community without a central leader. Anyone can connect their computer to the peer-to-peer network that manages the ledger. The XRP Ledger is the home of XRP, a digital asset designed to bridge the world's many currencies. The XRP Ledger is one part of the developing Internet of Value: a world in which money moves the way information does today [41].

XRP price today is $0.588816 USD. Its current rank in CoinMarketCap is #3, with a market cap of $26,693,547,203 USD. It has a circulating supply of 45,334,295,892 XRP coins.

A notable difference between ripple and bitcoin is mining, as all 100 billion XRP already exist and require no mining process. Ripple wallets require a minimum of 20 XRP for the initial deposit, and thus the malicious behavior of spam is avoided.

Ripple's blockchain looks more like a list, which consists of a ledger version or ledger snapshots. Therefore, we can refer to XRP genesis ledger. The ledger versions are ranked in order in the XRP Ledger.

A simple ledger version consists of the following elements:

- A ledger header: includes the ledger index, the hash value of the ledger data and other metadata.
- A transaction tree: includes the transactions incorporated in the previous block.
- A state tree: includes ledger objects, with data, balances, data for the current ledger

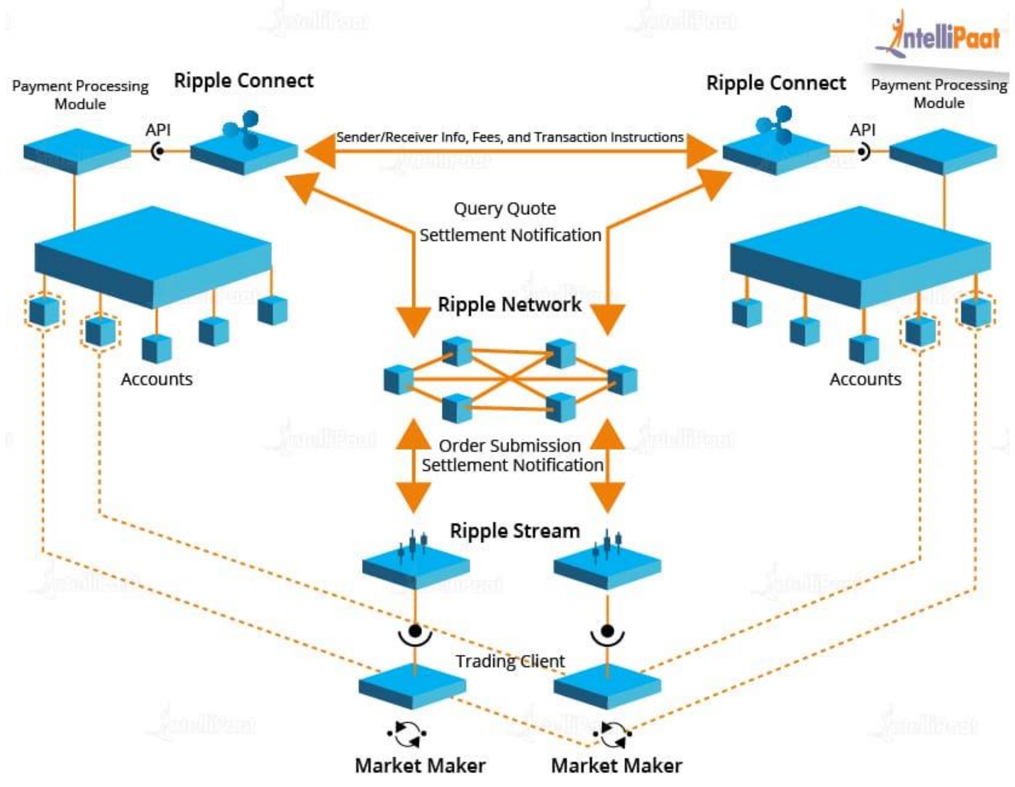


Image 9: Ripple Blockchain [86]

As described in [40], a Transaction is the only way to modify the XRP Ledger. Transactions are only final if signed, submitted, and accepted into a validated ledger version following the consensus process. Some ledger rules also generate pseudo-transactions, which aren't signed or submitted, but still must be accepted by consensus.

Every signed transaction has a unique "hash" that identifies it. The transaction hash can be used as a "proof of payment" since anyone can look up the transaction by its hash to verify its final status.

Transactions can do more than send money. For example, they are also used to rotate cryptographic keys. Transactions that fail are also included in ledgers because they modify balances of XRP to pay for the anti-spam transaction cost.

In the decentralized XRP Ledger, a digital signature proves that a transaction is authorized to do a specific set of actions. Only signed transactions can be submitted to the network and included in a validated ledger. A signed transaction is immutable: its contents cannot change, and the signature is not valid for any other transaction.

The Ripple network consists of a series of nodes, which execute the rippled protocol. There are two type of nodes, based on their activity:

Validating nodes: These nodes are responsible for processing the prospective transactions and deciding on the composition of the final ledger version. They also decide a few other parameters, such as the additional cost of transactions.

Stock/ Non-validating/ Tracking nodes: they provide protection against DoS attacks, access from rippled applications, and they save the XRP ledger history.

The XRP Ledger Consensus Protocol consists of 3 key components [5]:

- **Deliberation**, during which each node proposes a set of transactions for input into the ledger, based on suggestions received from other trusted nodes. When a node believes that it has several concordant proposals, it applies the corresponding transactions to the previous ledger, based on the protocol rules. It then applies validation to the new ledger.
- **Validation**, in which the nodes decide whether to fully validate a ledger, based on validations that have resulted from "trusted" nodes. When a quorum arises from validations for the same ledger, the specific ledger and its ancestors are considered fully validated, and their status is reliable and irrevocable.
- **Preferred branch**, in which the nodes decide the branch in which they want to work in ledger history. In the event of network congestion, the nodes may not have initially validated the same ledger for a given sequence number. In order to further progress and fully validate other ledgers, nodes utilize the ledger history of trusted validations to continue the process.

1.3.4 Dash

Dash is an altcoin that was forked from the Bitcoin protocol. It aims to ensure the possibility of direct transactions with low fees and to be a secure environment of complete privacy for users. It was launched in January 2014 by Evan Duffield. Its original name was “Xcoin”. The coin was treated with distrust from the beginning and that is why it was redesigned and re-released under the name “Darkcoin”. However, the suspicion continued to exist since its use was made on the dark net. Later, it rebranded again with the name Dash (digital cash).

Dash price today is $93.47 USD. The current CoinMarketCap ranking is #29, with a market cap of $921,426,640 USD. It has a circulating supply of 9,857,894 DASH coins. As in the case of Bitcoin, dash's total coins are the sum of a geometric series, but the final number of coins that will be issued is uncertain. The estimate of the max supply is 18,900,000 DASH coins.

Transactions allow users to spend duffs (10^-8 dash). Dash transactions are transmitted between network users in a serialized byte format, called raw format. The size of the transaction is affected by the number of input and output addresses involved in the transaction.

There are two important things about your Dash wallet:

- **Public Address**: something like the bank account number. This is randomly generated with a combination of letters and numbers, which you can share with anyone who wants to send your payment.
- **Private Key**: something like the ATM pin. It is also a random sequence of characters that you need to access your Dash wallet to make a transaction.

Dash was created in part to address two of bitcoin’s biggest problems:

- **Lack of privacy**: Bitcoin blockchain is completely public. This means that if you make a Bitcoin transaction, anyone in the world with internet access can find out the public address of the sender and receiver, the transaction's worth and the previous transactions in which that Bitcoin was involved. However, Dash offers a service called PrivateSend which adds privacy to transactions. Because of this, Dash transactions cannot be traced back, nor is the identity of users revealed to the world. Private transactions are facilitated by Master nodes.
- **Speed of transactions**: On average, it takes about 10 minutes for a Bitcoin transaction to get confirmed. Dash uses a special service called "InstantSend" to deal with this time problem.Using InstantSend, Dash transactions are almost instantly confirmed by the Masternode network.

Dash network transactions are secured using PoW mining. In this process, powerful computer processors look for solutions to a mathematically difficult problem defined by the hash algorithm X11, which was developed from its creator. Each user can take on the role of Master node, as long as they deposit the amount of 1000 Dash. The master node system uses PoS. Therefore, the Dash network combines both PoW and PoS.

On the Dash blockchain the system has miners and master nodes. Master nodes act as special servers that perform the critical functions on the Dash crypto network. They are responsible for PrivateSend services, InstantSend services and the governance and treasury system. These master nodes improve the security of the network and make sure that the transactions are as quick as cash transactions.

The reward is distributed as follows:

- 45% of the reward goes to miners
- 45% of the reward goes to master nodes
- 10% goes towards funding for further network improvements. The masternodes vote on these proposals and If approved, the amount raised in the budget is paid directly from the blockchain.

The average transaction fee for Dash is between $0.2 - $0.3, much cheaper than Bitcoin transaction fees that are in the range of $2 - $5.

As reported in [5], the Dash blockchain includes Dash ledger. Dash ledger corresponds to a classified and valid transaction file, and it is designed to prevent double spend attacks and keep the transactions unchanged. Each full node of the Dash network maintains a snapshot of the blockchain that contains only blocks that have been validated by this node. When a few nodes maintain the same blocks in their blockchain, then these nodes are in consensus. The validation rules used by these nodes to maintain this agreement are called consensus rules.

Each block in the Dash includes a block header. The bock header consists of:

- **Version**: the block version number
- **previous block header hash**: the hash value of the block header of the previous block
- **merkle root hash**: the hash value of the Merkle tree root used for blockchain transactions.
- **Time**: the block creation timestamp

- **nBits**: the target value for mining
- **Nonce**: the nonce factor for mining

1.3.5 Litecoin

Litecoin is a peer-to-peer Internet currency that enables payments to anyone in the world. Litecoin is an open source payment network that is not managed by any central authority. The cryptocurrency was created based on the Bitcoin protocol, and it was released via an open-source client on GitHub in 2011 by Charlie Lee, a former Google employee.

Litecoin price today is $71.60 USD. The current CoinMarketCap ranking is #6, with a market cap of $4,730,296,811 USD. It has a circulating supply of 66,065,398 LTC coins and a max. supply of 84,000,000 LTC coins.

Bitcoin and Litecoin seems to have many similarities. They are decentralized cryptocurrencies and they both use PoW systems. In both Litecoin and Bitcoin, the process of confirming transactions requires substantial computing power. Miners allocate their computing resources toward confirming the transactions of other users. In exchange for doing so, these miners are rewarded by earning units of the currency which they have mined. The reward halves every 840,000 blocks.

Although Litecoin was based on the Bitcoins protocol, they differ in many ways. The main differences between them are:

- Litecoin confirms transactions much faster than Bitcoin, as it processes a block every 2.5 minutes (Bitcoin needs 10 minutes).
- Litecoin uses the memory intensive Scrypt proof of work mining algorithm (Bitcoin uses SHA-256).

The Litecoin blockchain handles a higher volume of transactions than Bitcoin. The blocks are created faster, so the network supports more transactions, without the need to modify the software in the future. As a result, traders obtain transaction certificates in a shorter period of time, while at the same time being able to wait for more confirmations when selling items with a longer validity period.

1.3.6 Namecoin

Namecoin was the first fork of Bitcoin and still is one of the most innovative “altcoins”. It was introduced by Vinced on April 18, 2011. In June 2013, a service that links profile information with identities on the Namecoin blockchain named NameID. Namecoin is described as an experimental open-source technology which improves decentralization and security, and it was first to implement merged mining and a decentralized DNS. DNS (domain naming system) is the mechanism by which domain identities are linked with numerical IP addresses around the world. Namecoin can be used to record and transfer arbitrary names or keys securely. It can also attach data to these names.

Namecoin price today is $0.495294 USD. The current CoinMarketCap ranking is #626, with a market cap of $7,298,845 USD. It has a circulating supply of 14,736,400 NMC coins.

Namecoin has its own blockchain in which they subscribe:

- NMC transactions between users
- Domain names and to whom they belong

It is based on the code of bitcoin and uses the same PoW algorithm. Due to the common features with bitcoin, it is also limited to 21,000,000 coins, while 50 NMC coins result as a reward from mining.

However, it differs from bitcoin in crucial ways:

- Namecoin can store data within the blockchain transaction database.
- The merged mining feature allowed bitcoin and Namecoin to be mined simultaneously. Part of the reason for this feature was to incentivize miners to continue to mine both digital currencies at once, rather than switch back and forth to follow whichever became more profitable.
- Namecoin implemented a decentralized DNS. The Bitcoin model includes a peer-to-peer system in which participants continuously validate a series of transactions without central control. This model was implemented in DNS, modifying the Bitcoin protocol.

Comparison between Bitcoin and Namecoin:

Table 2: Namecoin vs Bitcoin

	Bitcoin	**Namecoin**
Definition	The first fully implemented peer-to-peer cryptocurrency protocol.	Is a type of Digital Cryptocurrency that acts as an alternative, decentralized DNS.
Inflation	Rate halves every 4 years	Rate halves every 4 years
Final Total	21 million	21 million
Method	Increasing difficulty per every 2016 blocks produced	Merge mined with Bitcoin
Algorithm	SHA-256	SHA-256
Mined using	GPUs, FPGA, or ASIC	Merge mined with Bitcoin

1.4 Smart Contracts

1.4.1 Introduction to Smart Contracts

A smart contract is a computer program or a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement. The objectives of smart contracts are the reduction of need for trusted intermediators, arbitrations and enforcement costs, fraud losses, as well as the reduction of malicious and accidental exceptions [75].

The term "Smart Contract" was coined by Nick Szabo in the 1994. Szabo defined it as “a set of promises, specified in digital form, including protocols within which the parties perform on the other promises.”

These can be used to create a wide range of Decentralized Applications (DApps) which can include games, digital collectibles, online-voting systems, financial products and many others.

Table 3: Traditional vs Smart Contracts [74]

Traditional Contracts	Smart Contracts
1-3 days	Minutes
Manual Remittance	Automatic Remittance
Escrow necessary	Escrow may not be necessary
Expensive	Fraction of the cost
Physical presence (wet signature)	Virtual presence (digital signature)
Lawyers necessary	Lawyers may not be necessary

According to [63] Smart contracts can:

- Function as 'multi-signature' accounts, so that funds are spent only when a required percentage of people agree
- Manage agreements between users, say, if one buys insurance from the other
- Provide utility to other contracts (like how a software library works)
- Store information about an application, such as domain registration information or membership records.

Bitcoin was the first to support basic smart contracts, in the sense that the network can transfer value from one person to another. The network of nodes will only validate transactions if certain conditions are met.

Ethereum is the biggest, most widely used and most developed smart contracts platform in the world. Ethereum replaces bitcoin’s restrictive language (a scripting language of a hundred or so scripts) with a language that allows developers to program their own smart contracts.

As described in [74],[70], Smart Contracts offer:

- Autonomy: Decentralized nature of the blockchain contracts underlying these smart contracts as well as the nature of the contract themselves means no outside party is required in the process. In other words, there’s no need for a broker, lawyer or other intermediaries to confirm, as you are the one making the agreement.
- Backup: Data storage devices can fail. Smart contracts duplicate all transactions so that all parties have a record of the transitions. The likelihood that all parties are going to suffer data storage failures is practically nonexistent.
- Trust: All documents are encrypted on a shared ledger. There’s no way that someone can say they lost it.
- Safety: Cryptography keeps your documents safe. Blockchain technology creates unalterable ledgers that provide definitive proof of transactions. It also relies on encryption while operating as a secure peer to peer transaction system.
- Speed: Speeds are much faster, as there is no need for a middleman. Moreover, smart contracts use software code to automate tasks, thereby shaving hours off

a range of business processes, such as paperwork to manually process documents.

- Cost savings: The automated process lowers costs significantly, as smart contracts do not need the presence of an intermediary.
- Accuracy: Automated contracts are not only faster and cheaper, but also avoid human mistakes and errors that come from manually filling out heaps of forms.

1.4.2 Ethereum Smart Contracts

A "smart contract" is simply a program that runs on the Ethereum blockchain and guarantee to produce the same result for everyone who runs them.

Smart contracts have a balance, and they can send transactions over the network. Although they are a type of Ethereum account, they're not controlled by a user. They are deployed to the network and run as programmed. By submitting transactions that execute a function defined on the smart contract, user accounts can interact with a smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code.

Table 4: Ethereum Smart Contracts-Characteristics [73]

Computer programs	Smart contracts are simply computer programs. The word “contract” has no legal meaning in this context.
Immutable	Once deployed, the code of a smart contract cannot change. Unlike with traditional software, the only way to modify a smart contract is to deploy a new instance.
Deterministic	The outcome of the execution of a smart contract is the same for everyone who runs it, given the context of the transaction that initiated its execution and the state of the Ethereum blockchain at the moment of execution.
EVM context	Smart contracts operate with a very limited execution context. They can access their own state, the context of the transaction that called them, and some information about the most recent blocks.
Decentralized world computer	The EVM runs as a local instance on every Ethereum node, but because all instances of the EVM operate on the same initial state and produce the same final state, the system as a whole operates as a single "world computer."

1.4.2.1 Smart Contracts and Vending Machines

Nick Szabo described a smart contract like a vending machine. Everyone who puts the correct amount of coins into the machine can expect to receive a product in exchange. Similarly, on Ethereum, contracts can hold value and unlock it only if specific conditions are met. In other words, with the right inputs, a certain output is guaranteed. Like a vending machine removes the need for a vendor employee, smart contracts can replace intermediaries in many industries.

Image 10: Smart Contract-Vending Machine [87]

1.4.2.2 Compasibility and Limitations

Smart contracts are public on Ethereum and can be thought of as open APIs. Anyone can call other smart contracts in his own smart contract to greatly extend what's possible. Contracts can even deploy other contracts [64].

Smart contracts cannot send HTTP requests, so they cannot by themselves get information about "real-world" events. It was designed this way, as relying on external information could jeopardize consensus, which is important for security and decentralization. This limitation is addressed using oracles, a bridge between the blockchain and the real world. They act as on-chain APIs you can query to get information into your smart contracts. This could be anything from price information to weather reports [64],[65].

1.4.2.3 Security issues

Issues in Ethereum smart contracts, in particular, include ambiguities and easy-but-insecure constructs in its contract language Solidity, compiler bugs, Ethereum Virtual Machine bugs, attacks on the blockchain network, the immutability of bugs and that there is no central source documenting known vulnerabilities, attacks and problematic constructs.

1.4.2.4 The Decentralized Autonomous Organization Attack

A blockchain-based smart contract is visible to all users of the blockchain. This means that bugs, including security holes, are also visible to all and these security problems may not be quickly fixed.

A characteristic example of this kind of attack was executed with success on The DAO (Decentralized Autonomous Organization) in June 2016. While programmers were working on fixing the problems and were trying to come to a solution that would gain consensus, an unknown attacker manage to drain US$50 million in Ether collected from the sale of its tokens.

DAO program had a time delay in place before the hacker could remove the funds; a hard fork of the Ethereum software was done to claw back the funds from the attacker before the time limit expired.

1.4.3 Smart Contracts Languages

Anyone can write a smart contract and deploy it to the network. You just need to learn how to code in a smart contract language and have enough ETH to deploy your contract. Deploying a smart contract is technically a transaction, so you need to pay your Gas (the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network) in the same way that you need to pay gas for a simple ETH transfer. Since each Ethereum transaction requires computational resources to execute, each transaction requires a fee. Gas refers to the fee required to successfully conduct a transaction on Ethereum. Gas costs for contract deployment are far higher, however.

As reported in [73], the currently supported high-level programming languages for smart contracts include (ordered by approximate age):

- **LLL**: A functional (declarative) programming language, with Lisp-like syntax. It was the first high-level language for Ethereum smart contracts, but is rarely used today.
- **Serpent**: A procedural (imperative) programming language with a syntax similar to Python. It can also be used to write functional (declarative) code, though it is not entirely free of side effects.
- **Solidity**: A procedural (imperative) programming language with a syntax similar to JavaScript, C++, or Java. The most popular and frequently used language for Ethereum smart contracts.
- **Vyper**: A more recently developed language, similar to Serpent and again with Python-like syntax. Intended to get closer to a pure-functional Python-like language than Serpent, but not to replace Serpent.
- **Bamboo**: A newly developed language, influenced by Erlang, with explicit state transitions and without iterative flows (loops). Intended to reduce side effects and increase auditability. Very new and yet to be widely adopted.

The two most active and maintained languages are Solidity and Vyper. **Solidity** is by far the most popular, to the point of being the de facto high-level language of Ethereum and even other EVM-like blockchains.

2. Mixing Services

2.1 Introduction in Mixing Services

Although crypto has appeared only a few years, it has already managed to change world finance, payments and purchases. However, design of cryptocurrency and blockchain networks are not able to support anonymous payments, as the identity of blockchain users can be exposed using simple deanonymization attacks.

Ensuring anonymity is very important in a blockchain network. If an attacker manages to find out the identity of a blockchain user, he can easily use specialized spyware, track all victim’s financial transactions and use that identity for theft, extortion or other crimes.

Mixing service is created in order to improve the anonymity of transactions. A mixer (or tumbler) is a service that mixes different streams of potentially identifiable cryptocurrency, and the level of anonymity provided by a mixing protocol can differ.

A mix-server or mixer is a network that shuffles a group of messages and passes them to the receiver in a permuted order. The primary purpose for such a mechanism is to provide “sender privacy”; that is, it ensures that the entity receiving the mix-server’s output cannot discern who transmitted a message. In some sense, the mix-server separates a set of senders from a single receiver and attempts to conceal the sender-message relationships [89].

The mixing protocol works as follows:

- The payer transfers the money to the mixing service.
- The mixing service mixes it with that of other users and transfers the mixed currency to the desired address.

Using this process, there is no connection between the original transaction and this address. The transaction is made up of many small partial payments spread over a longer period of time. The transaction amounts can be chosen at random. The mixing service usually charges a fee of between 0.25 and 3% of the amount to be mixed.

The user receives currency from other users during this process, meaning that they run the risk of receiving “dirty” currency. Currency used for illegal activities and could now connect payees to these activities. Mixing services are also used by criminals; for example, to launder stolen money. In some jurisdictions, mixing is therefore illegal under anti-structuring laws [90].

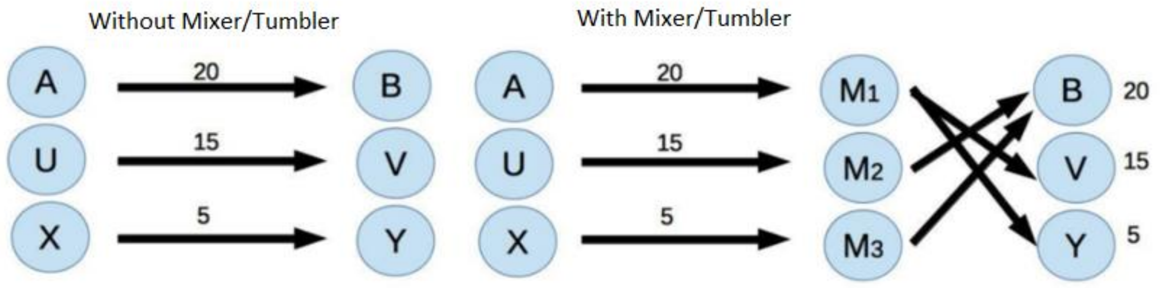


Image 11: Bitcoin Mixer [123]

2.1.1 Simple model of mixing

As described in [93], in all the examples of mixing models in this paper, we have 3 distinct entities: the mixer or the tumbling smart contract, a set of senders and a set of receivers. Alice is considered as the sender and Bob as the receiver. In a simple model of mixing a client (Alice) owns a number of coin in an address k_{in}, which is linkable with his real world identity. Alice wants to transfer some of his funds to a fresh address k_{out}, without linking k_{in} to k_{out} (and therefore Alice's real id), in exchange for a mixing fee. Alice will send some of his coins to a mix M, a for-profit entity which will hold Alice's funds in escrow for an agreed time period before sending an equal value to k_{out}. M is not required to have any real-world reputation or assets, only to maintain the same digital identity long enough to build a virtual reputation. Using this model, Alice may deal with two threats:

- Theft: Mixes send funds to fresh addresses with no transaction history, so it is possible for a malicious mix to send Alice's funds to its own secret address k_m instead of the requested one. There is no possibility for the observers to find out which address belongs to Alice or M, so Alice's complaints could be claimed as libelous. Mixes may try to undermine trust in their competitors through false accusations of theft, but there is no way to prove if these accusations are true, so it is hard to determine which mixes are honest.
- Deanonymization: The mix learns that the same party owns both addresses (k_{in}, k_{out}), so Alice's anonymity depends on the mix keeping this pairing secret forever. If mix share this information, the Alice's anonymity is undermined. Moreover, the mix could send coins in a non-random manner which reveals the connection to observers.

However, mixes might steal the user coins at any time or become a threat to the user's anonymity because the mix will know the internal mapping between the users and outputs.

2.1.2 Mixer protocol properties

The mixer protocol should offer the following properties, according to authors in [94] and [95]:

- **Accountability**: When a user sends funds to the mix, they should be confident that if a theft occurs, they can present a proof of the mix's misconduct.
- **Anonymity**: The user should be the only entity that knows the mapping from their input address to their output address. After a successful Bitcoin mixing transaction, honest participants' input and output addresses must be unlinkable
- **Resilience to Denial of Service Attacks**: The system should not be vulnerable to attacks by a small number of malicious parties that could potentially DoS the exchange.
- **Scalability**: The system should be efficient enough to be able to scale to a large number of users and large anonymity sets.
- **Incentive for Participation**: There should be a mechanism for the mix to collect mixing fees fairly that will incentivize it to provide the service. On the user end, the

service should come at a reasonable cost and should be convenient, so that a large number of users participate.

- **Backwards-Compatibility**: The system should be compatible with the current Bitcoin system to allow for practical integration.
- **Verifiability**: An attacker must not be able to steal or destroy honest participants' coins.
- **Robustness**: The protocol should eventually succeed even in the presence of malicious participants.
- **No Mixing Fees**: The protocol should not introduce additional fees specifically required for mixing. As every mixing transaction necessarily requires a Bitcoin transaction fee, the protocol must ensure that this transaction fee remains as low as possible.
- **Efficiency**: Even users with very restricted computational capacities should be able to run the mixing protocol. In addition, the users should not be required to wait for a transaction to be confirmed by the Bitcoin network during a run of the protocol, because this inherently takes several minutes.
- **Small Impact on Bitcoin**: The protocol should not put a large burden on the efficiency of the Bitcoin network. In particular, the size of the executed transactions should not be prohibitively large because all transactions must be stored in the blockchain and verified by all nodes in the network.

2.2 Cryptographic Techniques

2.2.1 Digital signature

As described in [89], a digital signature is a cryptographic technique, technologically equivalent to a hand-written signature or stamped seal. In many applications, digital signatures are used as building blocks in larger cryptographic protocols and systems.

A digital signature scheme is a triple of algorithms $(Gen, Sign, Verify)$ such that:

- Gen: The key generation algorithm. Take a security parameter 1^{λ} and output the pair (pk, sk).We call pk the public or verification key and sk the secret or signing key.
- $Sign$: The signing algorithm. When given a security parameter 1^{λ}, a signing key sk, and a message M, produce the digital signature σ of M.
- $Verify$: The verification algorithm. Take a verification key vk, a digital signature σ and a message m. Return $True = 1$ or $False = 0$ to indicate if the signature is valid.
- Gen and $Sign$ are PPT algorithms,$Verify$ is a deterministic polynomial-time algorithm.

Unforgeability, or the ability of a PPT adversary to construct a new legitimate message-signature pair, is the primary goal of digital signatures. The strongest attack against digital signatures is called the chosen method attack. An adversary has unrestricted access to a signing oracle in such an attack, which returns signatures to messages of the adversary's choice.

2.2.2 RSA Hashes

According to [105], RSA is one of the oldest public-key cryptosystem that is widely used for secure data transmission. Its algorithm was described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The acronym RSA comes from the surnames (Rivest–Shamir–Adleman). This algorithm allows message encoding, and it is used as a digital signature.

RSA is based on the practical difficulty of factorizing large prime numbers (today, usually in the range of 1024 to 2048 bits). Two keys are used, one public during encryption and one private for decryption. There are no published methods to defeat the system if a large enough key is used.

RSA is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, RSA is used to transmit shared keys for symmetric key cryptography, which are then used for bulk encryption-decryption.

2.2.2.1 RSA Key Generation

1. Selection of two large prime numbers p and $q, p! = q$. (Probabilistic algorithms are used to find prime numbers.)
2. Calculate $n = p.n$
3. Calculate Euler's function $\varphi(n) = (p-1)(q-1)$, (f must not share a factor with e)
4. Selection of a number $e > 1$ in a way that $e^{\varphi(n)} == 1\ (mod\ n)$ (Common options for e are 3, 7 and 2^{16} + 1. Small numbers lead to faster calculations but also to weaker security.)
5. Calculate d , in a way that $d ==\ e^{-1}\ (mod\ \varphi(n)\ )$

The public key is (n, e) and the private (n, d). Messages can be encrypted by anyone, via the public key, but can only be decoded by someone who knows the prime numbers.

2.2.2.2 RSA Encryption and Decryption

An encrypted message c is calculated as follows: $c = m^e\ mod\ n\ , m$ is the original message.

After receiving an encrypted message c ,to read the original message proceed to the following calculation:

$$m = c^d\ mod\ n\ \equiv (m^e)^d\ mod\ n\ \equiv m^{e.d}\ mod\ n$$

We know that $e.d\ \equiv\ 1\ (mod\ p-1)$ and $e.d\ \equiv\ 1\ (mod\ q-1)$, whenever with the small Fermat theorem, we have:

$$m^{e.d} \equiv m^1 \equiv m\ mod\ p-1$$

and

$$m^{e.d} \equiv m^1 \equiv m\ mod\ q-1$$

The numbers p and q are prime to each other, so using the Chinese Balance Theorem, we have:

$$m^{e.d} \equiv m \; mod \; n$$

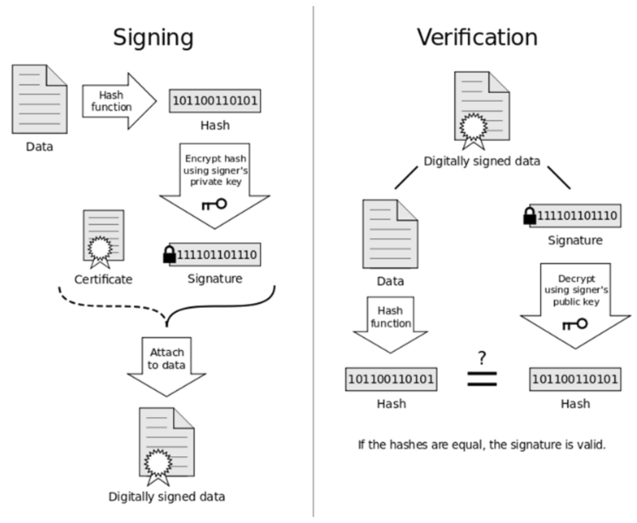


Image 12: RSA Hash [124]

2.2.2.3 RSA and Digital Signature

RSA allows digital signature of messages. If we want to send a signed message, we can do it in the following way (using the private key (n, d):

$$s = m^d mod \; n$$

The recipient of the message m and the signature s, calculates the value se thanks to the public key (n, e) and compares it with m. This solution, while working, is never used for security reasons. Instead of signing the message as is, it is preferable to use a hash function H:

$$s = H(m)^d mod \; n$$

The recipient uses the same method, as long as he knows which hash function was used.

2.2.2.4 RSA Security

Although the algorithm is considered secure when very large parameters are used, it can be exploited to expose major security flaws. Apart from this, to date, no one has proven that its safety is solely dependent on integer factorization.

There's also the possibility that someone could develop (or has already developed) an algorithm for factoring numbers in polynomial time.

Furthermore, since the algorithm employs an iterative function, a repeated encryption attack is possible. If we have the encrypted message and the public key used to encrypt it, we will proceed with the steps below:

We use the public key to encrypt the already encrypted message.Repeat the encrypting step until the result is the same as the first encrypted message.The decrypted text is found immediately after the encryption.

2.2.3 Zero-Knowledge Proof

As described in [89], a proof of knowledge protocol allows one party to convince another of the validity of a statement. In a zero-knowledge proof, this is accomplished without revealing any information beyond the legitimacy of the proof.

We have two parties, the prover P and the verifier V. P must convince V that he has some knowledge of a statement x without saying clearly what he knows. We call this knowledge a witness w. Both parties are aware of a predicate R that will attest to w being a valid witness to x. In general,

The predicate R is assumed to be polynomial-time computable: given a witness w for a statement x, one can efficiently test that $R(x, w) = 1$.

- The prover P has R, x, and w such that $R(x, w) = 1$. P wishes to prove possession of w by producing a proof of knowledge π.
- The verifier V has R, x, and π. In order for the above protocol to be useful in cryptographic applications, we can make the following assumptions.
- Given R, it is hard to find a corresponding w such that $R(x, w) = 1$.
- The prover P is reluctant to reveal w; otherwise the solution is trivial.
- The verifier V can efficiently check the validity of π.

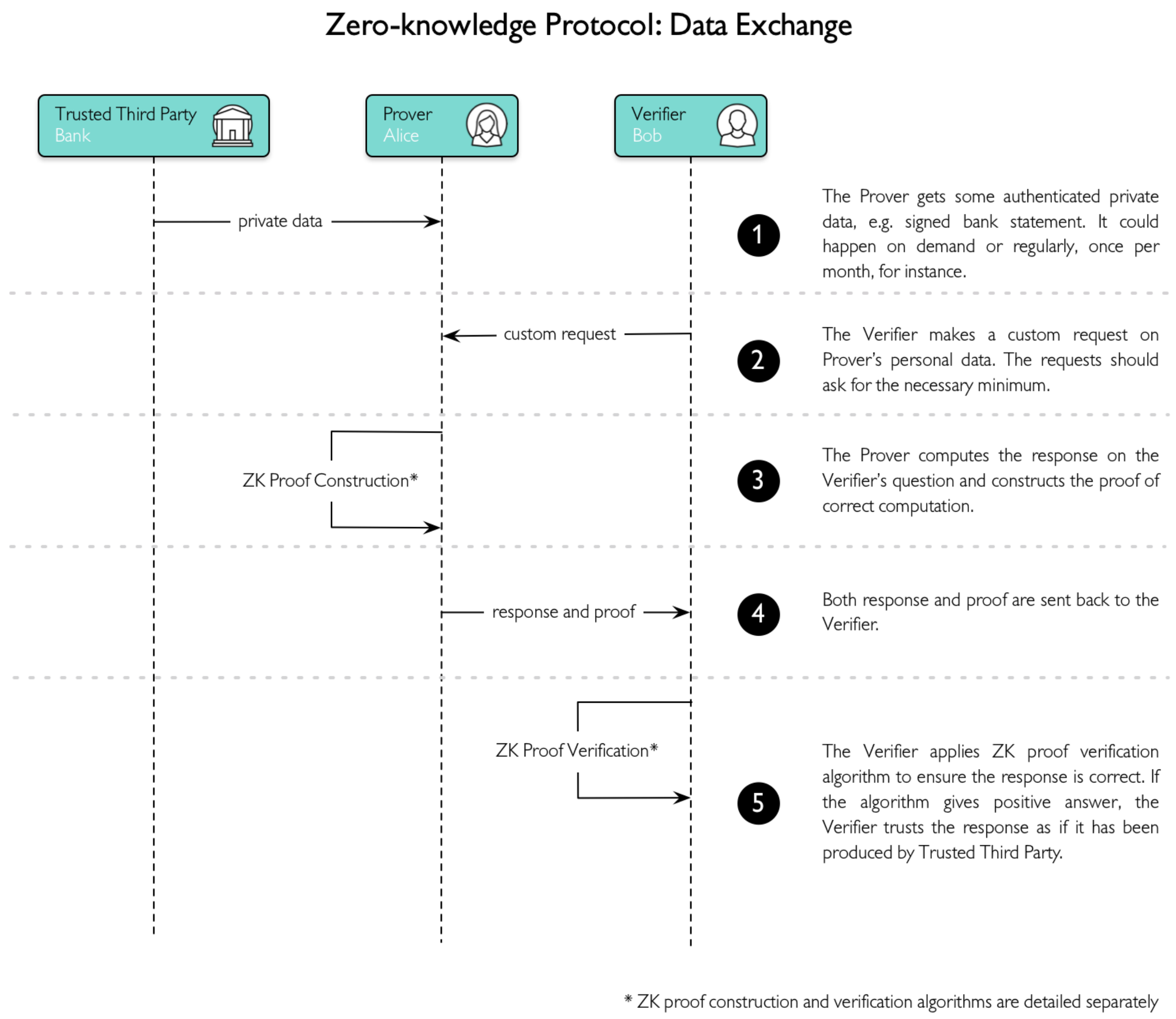

Zero-knowledge Protocol: Data Exchange
Trusted Third Party
Bank
Prover
Alice
Verifier
Bob
private data
1
The Prover gets some authenticated private data, e.g. signed bank statement. It could happen on demand or regularly, once per month, for instance.
custom request
2
The Verifier makes a custom request on Prover's personal data. The requests should ask for the necessary minimum.
ZK Proof Construction*
3
The Prover computes the response on the Verifier's question and constructs the proof of correct computation.
response and proof
4
Both response and proof are sent back to the Verifier.
ZK Proof Verification*
5
The Verifier applies ZK proof verification algorithm to ensure the response is correct. If the algorithm gives positive answer, the Verifier trusts the response as if it has been produced by Trusted Third Party.
* ZK proof construction and verification algorithms are detailed separately

Image 13: ZK Proof Protocol [125]

2.2.3.1 Zero-Knowledge Proof Types

There are two types of ZK Proofs, [102]:

- Interactive Zero-Knowledge Proof: In an interactive zero-knowledge proof, a prover uses a mathematical probability mechanism to convince the verifier of a particular fact.
- Non-Interactive Zero-Knowledge Proof (NIZKP): In non-interactive zero-knowledge proof an interactive process is not required. Meaning, the prover can issue all of the challenges at once, with the verifier(s) responding later.Collusion is less likely as a result of this. However, it requires additional machines and software to find out the sequence of experiments.

2.2.3.2 Zero-Knowledge Proof Properties

A zero-knowledge proof must satisfy three properties, [102]:

- **Completeness**: If the statement is true and all users obey the rules to the letter, the verifier would be persuaded without the need for outside assistance.

- **Soundness**: In any case, if the statement is false, the verifier would not be convinced, (even if the prover says that the statement is true for some small probability).
- **Zero-knowledge**: if the statement is true, no verifier learns anything other than the fact that the statement is true. In other words, just knowing the statement (not the secret) is sufficient is enough to conjure up a situation in which the prover knows the secret. This is formalized by demonstrating that every verifier has a simulator that can generate a transcript that "looks like" an interaction between the honest prover and the verifier in question, given only the statement to be proved (and no access to the prover).

The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero-knowledge.

Zero-knowledge proofs are not proofs in the mathematical sense of the term because there is some small probability, the soundness error, that a cheating prover will be able to convince the verifier of a false statement. In other words, zero-knowledge proofs are probabilistic "proofs" rather than deterministic proofs. However, there are techniques to decrease the soundness error to negligibly small values.

2.2.4 Ring Signature

Ring signatures were invented by Ron Rivest, Adi Shamir, and Yael Tauman Kalai, and introduced at ASIACRYPT in 2001. The name, ring signature, comes from the ring-like structure of the signature algorithm.

A ring signature is a type of digital signature that can be created and performed by any member of a group of users who has their own keys. A message with a ring signature receives an endorsement from someone in a specific group of people. It is impossible to decide which public key belongs to a single group member using a ring signature. In group signatures, users must register as members of the group and at any point in time its membership list is well-defined. There is no way to revoke an individual signature's anonymity in ring signatures, and any group of users may be used as a group without any additional setup. These are the characteristics that make a ring signature differ from a group signature.

2.2.4.1 Ring signature definition

In [96], ring is defined as a group of possible signers and the actual signer (the member who wants to sign some message using ring signature). Each of them has public/private key pairs, $(P1, S1), (P2, S2), \ldots, (Pn, Sn)$. The actual signer can compute a ring signature σ on a message m, on input $(m, Si, P1, \ldots, Pn)$. Anyone can check the validity of a ring signature given σ, m, and the public keys involved, $P1, \ldots, Pn$. If a ring signature is properly computed, it should pass the check. On the other hand, it should be hard for anyone to create a valid ring signature on any message for any group without knowing any of the private keys for that group.

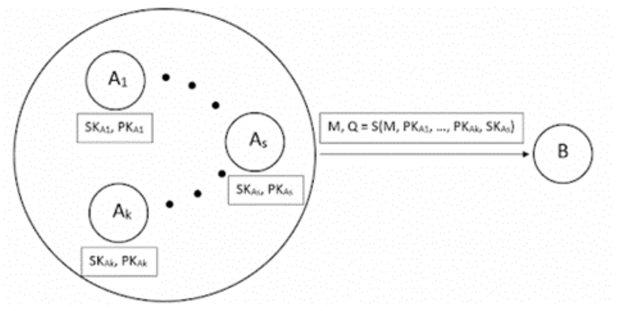


Image 14: Ring Signature [126]

So, ring signature scheme may be useful if it is necessary to provide signer's anonimity and his independence of other members, and thus ensure the integrity and authenticity of signed message.

The basic method, [96]:

The following is a breakdown of the basic method of creating a ring:

1. Generate encryption using $k = Hash(message)$
2. Generate a random value (u)
3. Encrypt u in order to give $v = Ek(u)$
4. For each participant (apart from the sender):
5. Calculate $e = si\hat{}\{Pi\}\ (mod\ Ni)$ and where si is the random number generated for the secret key of the ith party, and Pi is the public key of the party.
6. Calculate $v = v \oplus e$
7. For the signed party (z), calculate $sz = (v \oplus u)\hat{}d\ (mod\ Nz)$ and where d is the signing party's secret key

We will eventually end up with the signature $(v = Ek[u])$, and this finally completes the ring.

Regular ring signatures reveal only that the signer is part of the ring. They involve a tuple of algorithms ($Sign, Verify, Link$) such that ,[98]:

- The signing algorithm $Sign$: procedure of computing ring signature Q, which takes message M, public keys of ring members and signer's private key SKs as arguments.
- The verifing algorithm $Verify$: procedure of verifying the ring signature Q, which takes message M and ring signature Q as arguments; returns 1 if signature is correct and 0 otherwise.
- The linking algorithm $Link$: Linkable ring signatures use this additional algorithm, which is used to tell whether or not two signatures were produced by the same signer, without reveal their identity.

2.2.4.2 Ring Signatures and Security Proprties

The security properties of a linkable ring signature are, [99]:

- **Anonymity**: There is no way to connect a ring signature to a particular public key in the ring.
- **Unforgeability**: Without knowledge of a secret key corresponding to a public key included in the ring, a valid signature cannot be produced.
- **Exculpability**: Without control of this member's key, it is impossible to generate a valid signature that links to the signature of another ring member.
- **Linkability**: Within the same ring, any two signatures produced by the same signer are publicly linkable.

2.2.5 Blind signature

The idea of the blind signature was introduced by David Chaum. Blind signature is a form of digital signature in which the content of a message is blinded before it is signed. As a result, the signer would be unaware of the message's content. The signed message will be unblinded after that. At this moment, it is similar to a regular digital signature and the resulting blind signature can be publicly verified against the original unblinded message, [94].

A blind signature is a type of document authentication that enables a signer to authenticate a document without knowing anything about it. Unforgettability and blindness are the two primary goals of a blind signature, with blindness referring to the property of keeping certain details in a document hidden from the signer, [89].

2.2.5.1 Blind Signature Definition

Let e be a prime, M a string, n an RSA modulus, and H a hash function. A two-move blind signature scheme is a tuple of algorithms $(Blind, Sign, Unblind, Verify)$ such that, [89]:

- The blinding algorithm $Blind$: Given M, a random number $r \overset{r}{\leftarrow} Z_n$, and the public key $pk = (n, e)$, output $M' = r^e\, H(M) \bmod n$.
- The signing algorithm $Sign$: Given a blinded message $M'$and the secret key $sk = d$, output a digital signature $\sigma' = (M')^d \bmod n$.
- The algorithm $Unblind$: Given a signature σ' of a blinded message M', compute $\sigma = \sigma' r^{-1} \bmod n$.
- The verification algorithm $Verify$: For any (M, σ), test if $\sigma^e = (\sigma' r^{-1})^e = M' r^{-e} = H(M) \bmod n$. If equality holds, return $True = 1$; otherwise return $False = 0$.

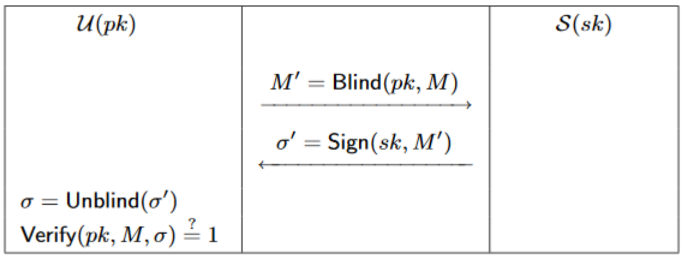

$\mathcal{U}(pk)$
$\mathcal{S}(sk)$
$M' = \text{Blind}(pk, M)$
$\sigma' = \text{Sign}(sk, M')$
$\sigma = \text{Unblind}(\sigma')$
$\text{Verify}(pk, M, \sigma) \overset{?}{=} 1$

Image 15: The Chaum blind signature generation [89]

2.2.5.2 RSA Blind Signature

A variety of popular public key signing schemes, such as RSA and DSA, can be used to enforce blind signature schemes. A blind signature scheme works as follows, [100]:

- A user wishes to receive a signature from the signer without the signer being aware of the contents of the message.
- The message is first "blinded", typically by combining it in some way with a random "blinding factor".
- The blinded message is sent to a signer, who uses a standard signing algorithm to create a "pre-signature" and sends it back to the recipient.
- The resulting message, as well as the blinding factor, can be later verified against the signer's public key.

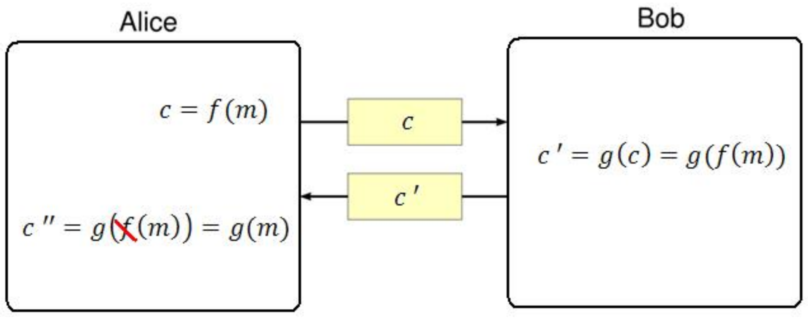


Image 16: RSA Blind Signature [127]

One of the simplest blind signature schemes is based on RSA signing. A traditional RSA signature is computed by raising the message m to the secret exponent d modulo the public modulus N. The blind version uses a random value r, such that r is relatively prime to N. r is raised to the public exponent e modulo N, and the resulting value $r^e mod\ N$ is used as a blinding factor.

The author of the message:

- computes the product of the message and blinding factor $m' \equiv mr^e (mod\ N)$
- sends the resulting value m' to the signing authority

Because r is a random value and the mapping $r \rightarrow r^e mod\ N$ is a permutation it follows that $r^e mod\ N$ is random too. This implies that does not leak any information about m.

The signing authority then:

- calculates the blinded signature s' as $s' \equiv (m')^d (mod\ N)$
- sends s' back to the author of the message

The author of the message:

- can then remove the blinding factor to reveal *s*, the valid RSA signature of *m*:

$s' \equiv s' r^{-1} (mod\ N)$

This works because RSA keys satisfy the equation $r^{ed} \equiv r(mod\ N)$ and thus

$s \equiv s' r^{-1} \equiv (m')^d r^{-1} \equiv m^d r^{ed} r^{-1} \equiv m^d r r^{-1} \equiv m^d\ (mod\ N),$

hence *s* is indeed the signature of *m*.

2.2.5.3 RSA Blind Signature Attacks

The RSA blinding attack, which can be used to trick RSA into decrypting a message by blind signing another message, is possible. Since the signing process is equivalent to decrypting with the signer's secret key, an attacker can provide a blinded version of a message m encrypted with the signer's public key, m' for them to sign. The encrypted message is typically some kind of sensitive information that the attacker noticed being sent encrypted under the signer's public key and is interested in learning more about, [100].

This attack works because the signer signs the message directly in this blind signature scheme. In an unblinded signature scheme, on the other hand, the signer will usually use a padding scheme. Instead of signing the message itself, the signer, for example, is signing the result of a cryptographic hash function applied to it. When unblinded, any padding scheme will produce an incorrect value, since the signer does not know the actual message. Due to this multiplicative property of RSA, it's never a good idea to use the same key for both encryption and signing, [100].

2.2.5.4 Blind Signature Properties

Blind signatures are often used in privacy-related protocols where the signer and message author are not the same individual. They can also be used to achieve the following requirements:

- **Unlinkability**: is beneficial in schemes that involve anonymity. It prevents the signer from associating the blind message he's signing with a later un-blind version that could be needed for verification.
- **Unforgeability**: An adversary cannot compute a valid signature on a new message given a set of blind signatures.

2.3 Mixing Strategies

A number of mixing strategies have been proposed and developed over the years. Bitcoin mixers or tumblers are software or services that mix your coins with other users’ coins to secure your privacy.

There are two major categories of mixers:

- centralized solutions
- decentralized solutions

2.3.1 Centralized Mixers

Centralized mixers are for-profit businesses that take your coins and give you various coins in exchange for a fee. When a large number of people use a mixing service, it becomes increasingly difficult for an outsider to link any of the "incoming" coins to any of the "outgoing" coins, providing users with privacy. To make a transaction, you have to enter your address in a form available on the mixing service platform and then submit your cryptocurrency to an address specified by the service, [106].

2.3.1.1 Centralized Mixer Algorithm

The algorithm steps are the following, [110]:

1. Alice sends cryptocurrency to an address specified by the service
2. The service takes a fee from your deposit, usually 1-3%
3. Alice specifies how much you want to withdraw (X) and during what period (T)
4. The service is then programmed to send transactions out for random amounts ($x < X$) such that $x_1 + x_2 + \dots\ x_n \leq X$, in random intervals ($t < T$) such that $t_1 + t_2 + \dots\ t_n \leq T$.

In other words, it'll split the withdrawal amount and send out small quantities over random periods of time, resulting in a final output that differs from the initial input.

The service collects and mixes various numbers of coins from various addresses before returning a random number of coins to each address. This random mixing and matching continues until you receive the total amount you requested in your wallet.

The more users who use a service, the more difficult it becomes to link any of the coins to a specific person.

2.3.1.2 Centralized Mixer Security

The security and privacy offered by these third party services are ambiguous. There are two main reasons why centralized mixers should be avoided:

- Users must trust the mixer with their personal information. Centralized mixers have access to your coins and IP addresses. Since the mixers know who sent and received which coins, they possibly keep a log or re-establish the ownership trail. All of this information may be exchanged (for example, by the law) or sold, and users' privacy would be compromised. In an ideal centralized mixer, all this information regarding the user is deleted.
- The mixer could refuse to refund the user's money, effectively stealing their coins.

2.3.2 Decentralized Mixers

Decentralized mixers are available on more advanced blockchain platforms, like those which support smart contracts. Decentralized mixers are designed to address the shortcomings of centralized mixing. These mixers are peer to peer mixing services available on more advanced blockchain platforms. Individuals form a group and pool their funds to make one significant transaction. The coins are returned to the pool members at random. Increasing the number of users in the pool will help increase the randomization. The most popular non-custodial mixers include Wasabi Wallet and Samourai Whirlpool, [106].

2.3.2.1 Decentralized Mixer Algorithm

The algorithm steps are the following, [108]:

1. Look for people who want to mix (group of like-minded people need to find each other).
2. Exchange input/output addresses.
3. Construct transaction (only a single person does this).

4. Send it around, collect signature (Before signing, each peer checks if her output is present: security property).
5. Broadcast the transactions.

2.3.2.2 Decentralized Mixer Security

Compared to centralized methods, decentralized mixers provide better security and anonymity guarantees. However, depending on the decentralized protocol used, they can face additional difficulties.

2.4 Security goals

As described in [111] and [99], the security goals are presented in terms of three goals: anonymity, availability, and theft prevention.

2.4.1 Anonymity

If an adversary cannot decide to whom honest senders are sending funds, sender anonymity is achieved, assuming that honest senders' deposits are indistinguishable.

If honest recipients' withdrawal transactions are indistinguishable, recipient anonymity is achieved.

2.4.1.1 Definition

Define $Adv_{mix,A}^{d-anon}(\lambda) = 2\Pr\left[G_{mix,A}^{d-anon}(\lambda)\right] - 1$ for $d \epsilon \{dep, with\}$, where these games are defined as follows:

$MAIN\ G_{mix,A}^{dep-anon}(\lambda)$	$MAIN\ G_{mix,A}^{with-anon}(\lambda)$
$(pk_i, sk_i) \overset{\$}{\leftarrow} KGen(1^\lambda) \forall i\ \in [n]$	$(pk_i, sk_i) \overset{\$}{\leftarrow} KGen(1^\lambda) \forall i\ \in [n]$
$PK_A \leftarrow \{pk_i\}_{i=1}^n; C, H_d, tumblers \leftarrow \emptyset$	$PK_B \leftarrow \{pk_i\}_{i=1}^n; C, H_d, tumblers \leftarrow \emptyset$
$b \overset{\$}{\leftarrow} \{0,1\}$	$b \overset{\$}{\leftarrow} \{0,1\}$
$(state, j, pk, i_0, i_1)$	$(state, j, pk, i_0, i_1) \overset{\$}{\leftarrow} A^{CORR,AD,HD,AW}(1^\lambda, PK_B)$
$\overset{\$}{\leftarrow} A^{CORR,AD,HD,AW}(1^\lambda, PK_A)$	$PK \leftarrow tumblers[j], keys_B$
$tx \overset{\$}{\leftarrow} Deposit(tumblers[j], sk_{A_{l_b}}, pk)$	$if\ \left(pk_{B_{l_0}} \notin PK\right) \vee \left(pk_{B_{l_1}} \notin PK\right) return\ 0$
$b' \overset{\$}{\leftarrow} A^{CORR,AD,HD,AW}(state, tx)$	$tx \overset{\$}{\leftarrow} Withdraw(tumblers[j], sk_{B_{l_b}})$
$\boldsymbol{return}\ b = b'$	$b' \overset{\$}{\leftarrow} A^{CORR,AD,HD,AW}(state, tx)$
	$if\ \left(pk_{l_b} \notin C\ for\ b \in \{0,1\}\ \right) return\ 0$
	$if\ ((j, l_b, \cdot) \in H_w for\ b\ \in \{0,1\})$
	$\boldsymbol{return}\ b = b'$

Then the tumbler satisfies sender or recipient anonymity if for all PPT adversaries A there exists a negligible function $v(\cdot)$ such that $Adv_{mix,A}^{dep-anon}(\lambda) < v(\lambda)$ or $Adv_{mix,A}^{with-anon}(\lambda) < v(\lambda)$ respectively.

2.4.2 Availability

It is essential for a coin mixer to provide availability, meaning that honest recipients can always withdraw their money from the mixer, even if senders and all but one recipients are compromised. Adversary A wins the availability security game if they manage to get the tumbler into a state where honest recipient cannot withdraw their funds.

2.4.2.1 Definition

Define $Adv_{mix,A}^{avail}(\lambda) = \Pr\left[G_{mix,A}^{avail}(\lambda)\right]$, where the game is defined as follows:

```
$MAIN\ G_{mix,A}^{avail}(\lambda)$
$(pk_i, sk_i) \overset{\$}{\leftarrow} KGen(1^\lambda) \forall i \in [n]$
$PK_B \leftarrow \{pk_i\}_{i=1}^{n}; C, H_w \leftarrow \emptyset$
$(l, j) \overset{\$}{\leftarrow} A^{CORR,AD,HD,AW}(1^\lambda, PK_B)$
$b \longleftarrow VerifyWithdraw(tumblers[j], Withdraw(sk_l))$
$if\ (pk_l \in C) \vee ((j, l, \cdot) \in H_w) return\ 0$
$\boldsymbol{return}\ (b = 0) \wedge (pk_l \in tumblers[j], keys_B)$
```

Then the tumbler satisfies availability if for all PPT adversaries A there exists a negligible function $v(\cdot)$ such that $Adv_{mix,A}^{avail}(\lambda) < v(\lambda)$.

2.4.3 Theft prevention

It should be ensured that neither coins can be withdrawn twice, nor withdrawn by anyone other but the intended recipient.

2.4.3.1 Definition

Define $Adv_{mix,A}^{theft}(\lambda) = \Pr\left[G_{mix,A}^{theft}(\lambda)\right]$, where the game is defined as follows:

```
$MAIN\ G_{mix,A}^{theft}(\lambda)$
$(pk_i, sk_i) \overset{\$}{\leftarrow} KGen(1^\lambda) \forall i \in [n]$
$PK_B \leftarrow \{pk_i\}_{i=1}^{n}; C, H_w, contract \leftarrow \emptyset$
$(tx, j) \overset{\$}{\leftarrow} A^{CORR,AD,HD,AW}(1^\lambda, PK_B)$
$if\ (tumblers[j].keys_B \not\subset PK_B \backslash C\ )\ return\ 0$
$\boldsymbol{return}\ VerifyWithdraw(tumblers[j], tx)$
```

Then the tumbler satisfies theft prevention if for all PPT adversaries A there exists a negligible function $v(\cdot)$ such that $Adv_{mix,A}^{theft}(\lambda) < v(\lambda)$.

2.5 Mixing Protocols

In this section, both centralized and decentralized mixing protocols will be presented.

2.5.1 Centralized Mixing Protocols

2.5.1.1 Mixcoin Protocol

MixCoin is a third-party mixing protocol designed to make anonymous Bitcoin and other cryptocurrency payments possible. It is completely compatible with Bitcoin and provides an anonymity set of all other users mixing coins contemporaneously. The Mixcoin makes use of the emergent phenomenon of currency mixes, in which a consumer sends a certain number of coins to a third-party mix through a standard-sized transaction and receives the same number of coins back from the mix sent by another user, providing strong anonymity from external entries. To prevent other users within the mix from theft and disrupting the protocol, Mixcoin uses a reputation-based cryptographic accountability technique [93].

The aim of Mixcoin is to provide a protocol for mixing with accountability. Before the mixing process, the mix signs a warranty agreement that allows the user to show beyond a doubt if the mix has misbehaved. If the user pays funds to an escrow address specified by the mix by a certain time, the mix should send an equivalent amount of funds to the user's output address before the deadline. Dishonest mixes' reputations will soon be damaged, and they will lose business, [93].

2.5.1.1.1 How Mixcoin works

According to [93], a warranty-signing key KM_i represents one of the multiple mixes M_i. Mixes are motivated to build and maintain a reputation in KM_i because they are for-profit enterprises, so it must be used consistently. Alice doesn't need to keep a long-term public key or a public reputation, but she does need to be able to communicate with the mix anonymously and privately. In reality, this would most likely be accomplished by mixes that run a dedicated Tor hidden service, but this is out of the scope of the Mixcoin protocol itself.

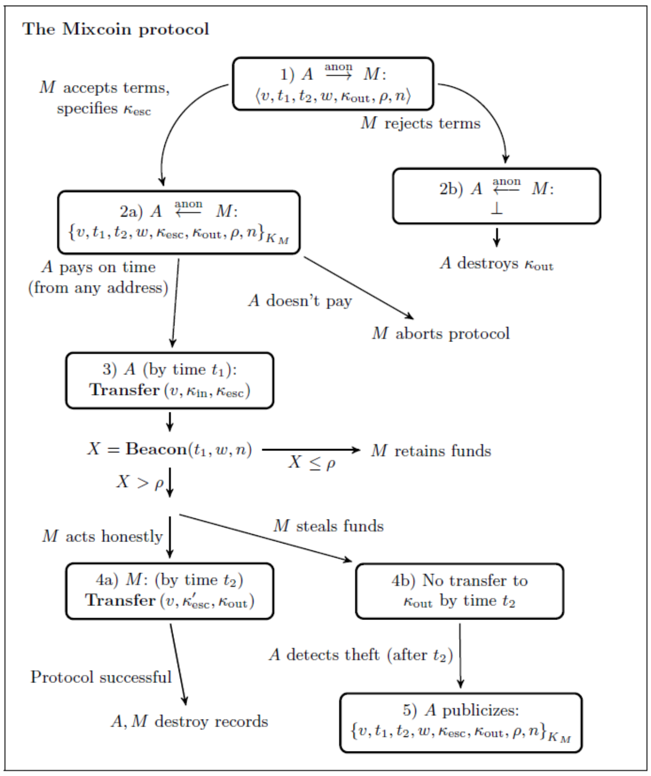
As referred in [93], the parameters used in this protocol are the following:

- v : the value (chunk size) to be mixed. Each user sends a fixed quantity v of coins to the mix
- t_1: the deadline by which Alice must send funds to the mix
- t_2: the deadline by which the mix must return funds to Alice
- k_{out}: the address where Alice wishes to transfer his funds
- ρ : the mixing fee rate Alice will pay
- n : a nonce, used to determine payment of randomized mixing fees
- w the number of blocks the mix requires confirming Alice's payment
- k_{esc}: a fresh escrow address
- Beacon: a pseudorandom function, which depends on a public source of randomness, using the Bitcoin block $t_1 + w$ plus the nonce n, and compared to the fee rate ρ.

Both the mix's escrow address k_{esc} and Alice's output address k_{out} should be fresh addresses created specifically for this mixing.

Mixing fees are randomized with probability p the mix retains the entire value v as a fee, and with probability $(1 - p)$ takes no fee at all. This mixing fee calculation, according to the authors, can achieve the strongest anonymity properties rather than having fixed mixing fees. If the mixing fees are high enough, a reasonable mix seeking to maximize profits would not risk being caught if they cheat. A warranty showing their dishonesty would damage their reputation and jeopardize their business model.

Alice will have to pay transaction fees to Bitcoin miners in addition to mixing fees to ensure her transactions are included in the blockchain.



The Mixcoin protocol
1) $A \overset{\text{anon}}{\longrightarrow} M$: $\langle v, t_1, t_2, w, \kappa_{\text{out}}, \rho, n \rangle$
M accepts terms, specifies κ_{esc}
M rejects terms
2a) $A \overset{\text{anon}}{\longleftarrow} M$: $\{v, t_1, t_2, w, \kappa_{\text{esc}}, \kappa_{\text{out}}, \rho, n\}_{K_M}$
2b) $A \overset{\text{anon}}{\longleftarrow} M$: $\perp$
A destroys κ_{out}
A pays on time (from any address)
A doesn't pay
M aborts protocol
3) A (by time t_1): **Transfer** $(v, \kappa_{\text{in}}, \kappa_{\text{esc}})$
X = **Beacon**(t_1, w, n)
$X \leq \rho$
M retains funds
$X > \rho$
M acts honestly
M steals funds
4a) M: (by time t_2) **Transfer** $(v, \kappa'_{\text{esc}}, \kappa_{\text{out}})$
4b) No transfer to κ_{out} by time t_2
Protocol successful
A, M destroy records
A detects theft (after t_2)
5) A publicizes: $\{v, t_1, t_2, w, \kappa_{\text{esc}}, \kappa_{\text{out}}, \rho, n\}_{K_M}$

Image 17: The Mixcoin Protocol [93]

2.5.1.1.2 Mixcoin Security

The mapping from a user's input to output address is visible to the mixing server in the Mixcoin protocol, and it must be trusted to keep this information private. There's no way to verify that a mix isn't storing information about its users to deanonymize them. The mix could expose this information at any time in the future, jeopardizing the privacy of its users.

If a user wishes to stay anonymous in the event that the mix is compromised, he or she can mix coins in several rounds of separate mixes, which will be time-consuming and expensive in terms of mixing fees.

Focusing on an attacker who wants to gain as much information as possible about the anonymity set of possible pre-mixing input addresses which may have been the source of the funds held by a final output address k_{out}, there are the following threats according to the authors:

- The passive adversary's view with mix indistinguishability
- Active adversaries and distinguishable mixes
- Anonymity sets and mix delay

- Mixing multiple chunks
- Convergence of free parameters
- Side channels

2.5.1.2 Blindcoin Protocol

To provide internal unlinkability (i.e., stopping the mix from learning input-output linking) in MixCoin, authors in [94] proposes BlindCoin, a protocol that extends the MixCoin protocol by using blind signatures to generate user inputs and cryptographically blinded outputs called blinded tokens. However, BlindCoin needs two additional transactions to publish and redeem the blinded tokens in order to achieve this internal unlinkability, and the risk of theft from the mix remains.

Many characteristics of the Mixcoin scheme are carried over to the Blindcoin protocol. Mix accountability is a property inherited from Mixcoin, and it is defined as the property that if a mix violates the protocol and steals the funds of a user, their cheating can be proven. As a result, the mix provides a warranty to the user in order for him to trust it. The warranty offered in Blind coin is more complex than in Mixcoin. Not only does BlindCoin provide properties inherited by Mixcoin, such as accountability, anonymity, resilience to Denial of Service Attacks and scalability, but also it provides Backwards-Compatibility and Incentive for Participation.

Blindcoin protocol includes a third entity, the public log, in addition to the two entities of the Mixcoin protocol (user and mixing server). The public log is a public, append-only log used for third party verification purposes. When a party breaks policy, the public log would provide enough information to incriminate the misbehaving party. Anyone can make a post in this log, which has a timestamp and cannot be deleted. One possible way to implement this would be within the Bitcoin blockchain. To send a message, the sender only needs to transfer a small sum of coins to an address that corresponds to the message. If the user wishes to maintain his anonymity, the coins must be sent from an address that cannot be connected to the user's personal information.

2.5.1.2.1 How Blindcoin works

This protocol works as follows, according to [94]:

- the mix announces mix parameters,
- the user opts into the mix,
- the mix sends a partial warranty back to the user, which states that “if the user pays, the mix will publish the signed blinded token to the public log by the warranty deadline”. The mix waits until the user has paid to publish the token to the public log so that it can make sure that only valid, paid-for tokens are in the log.
- the user transfers funds,
- the mix completes the warranty by posting to the public log,
- the user unblinds the output address,
- and finally, the mix transfers funds to the output address.

The reason that the token must be published to the log is so that third party verifiers can check that the mix has acknowledged a user's payment. If the mix fails to publish the signed blinded token to the public log by the agreed-upon deadline, the user can present an evidence of misconduct, which could damage the mix’s reputation.

The mix could theoretically cheat when determining which output addresses to send funds to. If users have correctly followed the protocol and published their signed, unblinded tokens to the public log, then any party can see that the mix has signed the token. Furthermore, after the mix computes Beacon to determine the random collection of fees, it is possible for any party to check that the mix has correctly computed the function, since Beacon is public. If the mix fails to send funds to an output address that has passed Beacon, then the user present proof of wrongdoing once more.

The parameters/functions used are:

k_{in}: the address from which the Alice pays, possibly linked to Alice's true identity.

k_{out}: the address to which the user wishes funds transferred.

k_{esc}: an escrow address, unique for each user, that M provides for Alice to pay.

k'_{esc}: an escrow address that M uses to pay to k_{out}.

A': an anonymous identity that Alice can use to post to the public log.

M_{pub}: the public key of M.

M_{priv}: the private signing key of M.

A_C: a secret commitment/encryption function of Alice.

$A_{C'}$: the inverse of A_C.

ω: the number of blocks M requires confirming Alice's payment.

n: a per-user nonce, used to determine payment of randomized mixing fees.

v: the value (chunk size) to be mixed.

ρ: the mixing fee rate Alice will pay.

T the token, which is the triple (k_{out},n).

t_1: the time by which Alice must v BTC to k_{esc} in order to participate in the mix.

t_2: the time by which M must post the token T to the public log.

t_3: the time by which A' must unblind the output address via the public log.

t_4: the time by which the mix must transfer v BTC to k_{out}.

D the mix parameters, a tuple $\{t_1, t_2, t_3, t_4, v, \omega, \rho\}$.

$Beacon$: the beacon function, a publicly verifiable random function.

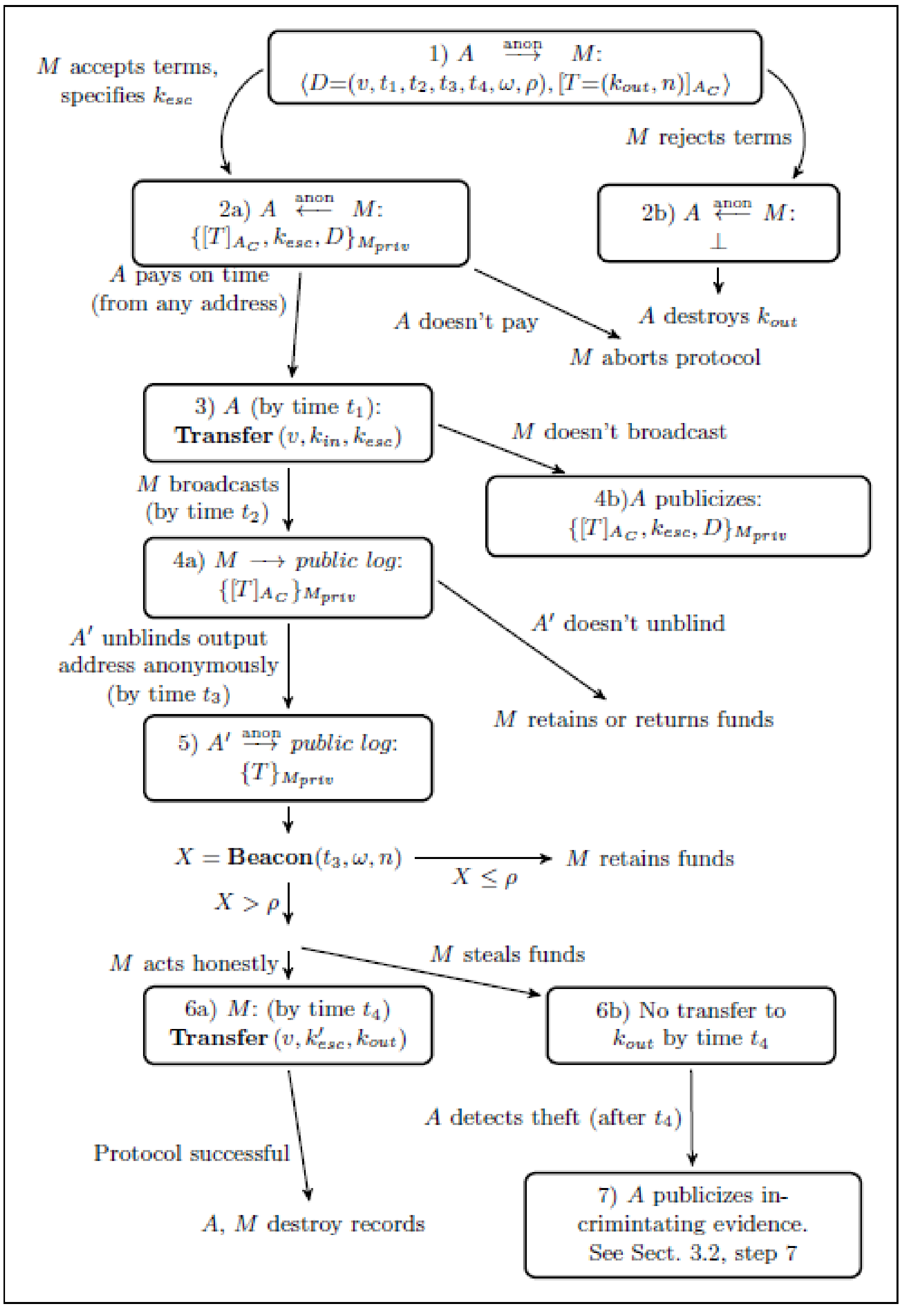

M accepts terms, specifies k_{esc}
1) $A \xrightarrow{anon} M$: $\langle D=(v,t_1,t_2,t_3,t_4,\omega,\rho),[T=(k_{out},n)]_{A_C}\rangle$
M rejects terms
2a) $A \xleftarrow{anon} M$: $\{[T]_{A_C}, k_{esc}, D\}_{M_{priv}}$
2b) $A \xleftarrow{anon} M$: $\perp$
A pays on time (from any address)
A doesn't pay
A destroys k_{out}
M aborts protocol
3) A (by time t_1): **Transfer** (v, k_{in}, k_{esc})
M doesn't broadcast
M broadcasts (by time t_2)
4b) A publicizes: $\{[T]_{A_C}, k_{esc}, D\}_{M_{priv}}$
4a) $M \longrightarrow$ *public log*: $\{[T]_{A_C}\}_{M_{priv}}$
A' doesn't unblind
A' unblinds output address anonymously (by time t_3)
M retains or returns funds
5) $A' \xrightarrow{anon}$ *public log*: $\{T\}_{M_{priv}}$
$X = \textbf{Beacon}(t_3,\omega,n)$
$X \leq \rho$
M retains funds
$X > \rho$
M acts honestly
M steals funds
6a) M: (by time t_4) **Transfer** (v, k'_{esc}, k_{out})
6b) No transfer to k_{out} by time t_4
A detects theft (after t_4)
Protocol successful
A, M destroy records
7) A publicizes incrimintating evidence. See Sect. 3.2, step 7

Image 18: The Blindcoin Protocol [94]

2.5.1.2.2 Blindcoin Security

Even in the case of malicious mix, anonymity is guaranteed in Blindcoin. The following threat models are solved by Blindcoin: global passive adversary, active attacker who can compromise some subset of the input/output address pairs for a particular mixing operation, mix is the adversary.

One property that Mixcoin provides is mix indistinguishability. Mixcoin is the only mixing service that offers this guarantee, and no other service offers it. Against a global passive adversary, this property extends a user's anonymity set to all users participating in different mixes that use the same parameters for mixing.

Side channels are another problem that Blindcoin has to deal with. Side channels can leak user information, which can lead to deanonymization and address linkage if not handled properly. Timing, precise values, network layer information, and interactions with the public log are all examples of side channels.

2.5.1.3 Tumblebit

In [112], authors present Tumblebit. Tumblebit is an unidirectional unlikable payment hub that enables quick, anonymous, off-blockchain payments through an untrustworthy intermediary, the Tumbler T, while remaining completely compliant with Bitcoin's protocol. Every payment made via TumbleBit is backed by bitcoins and comes with a promise that Tumbler cannot violate anonymity, or steal bitcoins, or “print money” by issuing payments to itself. This is achieved by cryptographic techniques.

TumbleBit enables a payer to send fast off-blockchain payments to a set of payees of his choice. The lack of coordination among Tumblebit users allows the device to scale. Alice just has to deal with Bob and Tumbler.

TumbleBit helps to scale the volume and velocity of bitcoin-backed payments by handling payments off the blockchain. TumbleBit payments are completed in seconds and are sent off-chain via the Tumbler T. In classic tumbler mode, performance is restricted only by the time it takes for two blocks on the blockchain to be validated and for transactions to be confirmed; currently, this takes about 20 minutes.

TumbleBit's problem isn't a lack of liquidity, but rather a surplus of it. TumbleBit requires the Tumbler Hub to escrow many bitcoins in order to function. Some people would be unable to use it if it cannot. As a result, TumbleBit's growth is essentially a direct Bitcoin buying pressure.

2.5.1.3.1 How Tumblebit works

Authors in [112] present the following example: Alice wants to pay Bob without being watched by observers or the TumbleBit service itself. This is accomplished with the aid of a bitcoin-related service that solves RSA puzzles. Cracking these cryptographic puzzles is currently impossible because finding the corresponding private key is as difficult as cracking an RSA public key. Furthermore, the puzzles use RSA blinding, which prevents the solution from being linked to a specific puzzle. It only knows that one of the puzzles it released was solved, but not which one.

The payer, Alice, creates a bitcoin payment channel with the TumbleBit hub. The hub gets paid after Alice presents a solution to her cryptographic problem. The Puzzle-Solver Protocol is the term given to this arrangement. The protocol enforces that the server should not collect until it has presented a solution, and if it does present a solution Alice must pay.

Bob, the payee, on the other end, has entered into a related agreement with the hub. The tumbler promises to pay Bob if a solution to the RSA puzzle is presented. This is known as the Puzzle-Promise Protocol.

We can break the connection between the puzzle solver and the puzzle maker by combining the blinding properties of RSA puzzles with enough decoy puzzles. Because of the properties of RSA encryption, the TumbleBit server knows the answer is right, but it can't tell which puzzle it's solving, effectively obfuscating the payer-payee relationship.

The protocol works well during the epoch where there are multiple parties using the service, after which it aggregates all transactions into a single transaction that redeems balances to payees. To avoid tracing analysis based on transaction amounts, TumbleBit demands that the transaction amounts be equal.

It's worth mentioning that RSA cryptography requires the use of two extremely large prime numbers, which is considered a secure setup but may be a problem for the most paranoid.

2.5.1.3.2 Tumblebit Phases

Alice A pays Bob B by providing B with the solution to a puzzle.

The puzzle z is generated through interaction between B and T, and solved through an interaction between A and T.

Each time a puzzle is solved, 1 bitcoin is transferred from Alice.

The protocol proceeds in three phases, [112]:

1. **Escrow Phase**: Each payer Alice A opens a payment channel with the Tumbler T by escrowing Q bitcoins on the blockchain, during the on-blockchain Escrow Phase. Each of the payees, Bob B, establishes a channel with T. T escrows Q bitcoins on the blockchain, while B and T participate in a puzzle-promise protocol that creates up to Q puzzles for B.
2. **Payment Phase**: Each payer A makes up to Q off-blockchain payments to any set of payees, in the off-blockchain Payment Phase. A interacts with T to learn the solution to a puzzle B given, in order to make a payment.
3. **Cash-Out**: The Cash-Out Phase closes all payment channels. Each payee B uses his Q' solved puzzles (TumbleBit payments) to create an on-blockchain transaction that claims Q' bitcoins from T 's escrow. Each payer A also closes her escrow with T, recovering bitcoins that were not used in a payment.

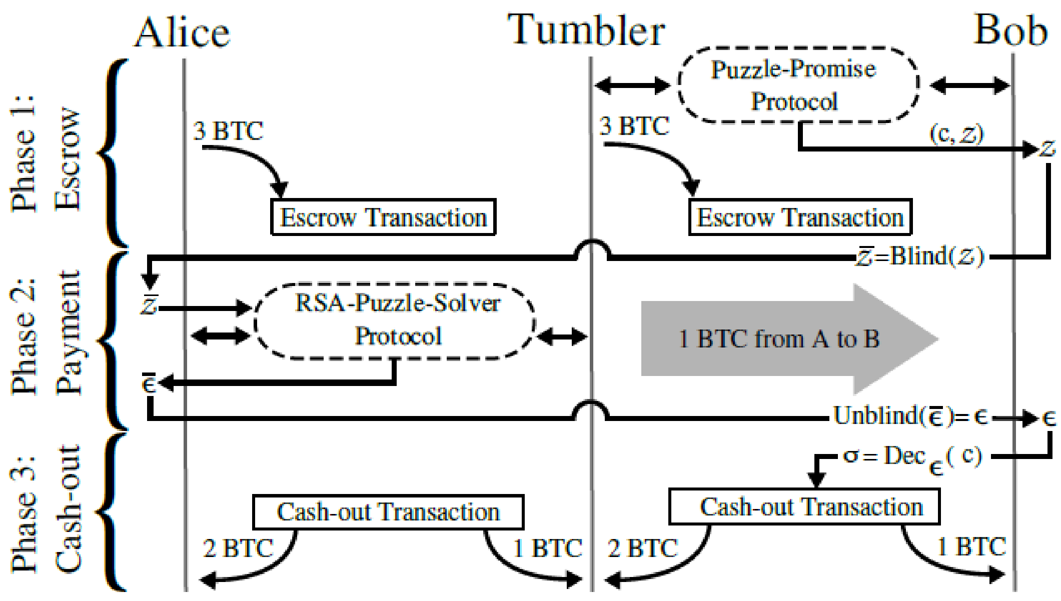


Image 19: Tumblebit Phases [112]

Mix services aim to provide anonymity in such a way that it's difficult to figure out who sent bitcoins to whom, but it's also critical to provide anonymity inside the mix service itself. TumbleBit offers k-anonymity for a set period of time and no one, not even the Tumbler T, can link one of the k transfers that were successfully completed during this period to a particular payer and payee pair
(A, B), [112].

2.5.1.3.3 Tumblebit Security

Tumblebit is vulnerable to the following attacks, [112]:

- **Alice/Tumbler collusion**: The concept of unlinkability assumes that the Tumbler does not collude with other TumbleBit users. Collusion between the Tumbler and Alice, on the other hand, can be used in a ceiling attack.
- **Potato attack**: External information is not included in the definition of unlinkability.
- **Intersection Attacks**: While this concept of k-anonymity is widely used in Bitcoin tumblers, it has the following flaw. Any adversary who watches the transactions on the blockchain within a given epoch will figure out who the payers and payees were for that epoch. Then, using this information, users can be de-anonymized through epochs.
- **Dos and Sybil Attacks**: TumbleBit defends itself from DoS and Sybil attacks by charging transaction fees As an incentive to validate transactions, any Bitcoin transaction can involve a transaction fee that is charged to the Bitcoin miner who confirms the transaction on the blockchain.

2.5.2 Decentralized Mixing Protocols

2.5.2.1 Mixing protocols based on Smart Contracts

As referred in [119], anonymous payment solutions will support the Ethereum ecosystem. While not everyone wants to use this currency for illegal purposes, some people always choose to have more privacy. Unfortunately, in the Ethereum ecosystem, this is currently very difficult to do.

The goal is to make Ethereum mixing as decentralized as possible, to bring this idea into perspective. Mixing services are operated by one or more operators in the Bitcoin world.

When using such a service, the sender must have complete faith in the mixer operator not to steal their money. Although this is typically not a big problem, it does trigger some confidence issues along the way. For mixing services, a decentralized solution makes a lot of sense, but it's also more difficult to incorporate.

The key concept is to provide decentralized Ethereum mixing services using smart contracts. The contract allows several parties to deposit and withdraw the same sum of money using various anonymous addresses.

All of the anonymous addresses that would obtain funds from the mixing contract had to have a small ETH balance to begin with. This is a downside because it means that a link to these anonymous addresses must already exist. Alternatively, mining incentives or faucet payouts may be sent to these addresses, preserving some anonymity.

Anyone wishing to obtain funds to an anonymous address would need to deposit 1 ETH via the smart contract in the current iteration. This is done to ensure that the user has control over the private key associated with the anonymous address. The contract will not execute if there are more anonymous addresses than person 1 ETH deposits. If this is not the case, however, the anonymous addresses can withdraw their claim and forfeit their registration deposit.

2.5.2.2 Coinjoin Protocol

Greg Maxwell, a core Bitcoin developer, proposes CoinJoin to perform mixing in a perfectly compatible manner with Bitcoin, while ensuring that even a malicious mixing server cannot steal coins. CoinJoin is a decentralized protocol that allows several parties to sign an agreement to mix their coins in a Bitcoin transaction where the output of the transaction leaves the participants with the same number of coins, but the addresses have been mixed to make external tracking difficult. Since a user can only supply their signature if they agree to the transaction, users remain anonymous and their funds cannot be stolen.

When Bitcoin users conduct transactions with each other, CoinJoin maintains their privacy. It's a way of merging multiple Bitcoin transfers from multiple spenders into a single transaction to make it more difficult for third parties to figure out who paid whom.

In other words, this protocol aims to enhance privacy by merging multiple users' inputs into a single transaction and discarding unnecessary data. For a single Bitcoin transaction, many users join forces and merge their inputs. All the signatures for each of the inputs are entirely separate, so a single user does not have to hold the private keys. The transaction generates a number of outputs, all of which hide the source of the coins. Some mixers design the CoinJoin in such a way that no one, including themselves, can find out what went where in the transaction.

2.5.2.2.1 How Coinjoin works

A user who wants to use CoinJoin in a Bitcoin transaction must first find another user who wants to mix coins and then start a joint transaction with them. The address a Bitcoin is sent from is referred to as an input.

Consider the following simultaneous transactions: A purchases an item from B, C purchases an item from D, and E purchases an item from F. Without CoinJoin, each input-output match will result in three different transactions on the public blockchain ledger. With CoinJoin, only one single transaction is recorded. The ledger would show that bitcoins were paid from A, C, and E addresses to B, D, and F. An observer cannot be certain who sent bitcoins to whom due to the masking of all parties' transactions, [115].

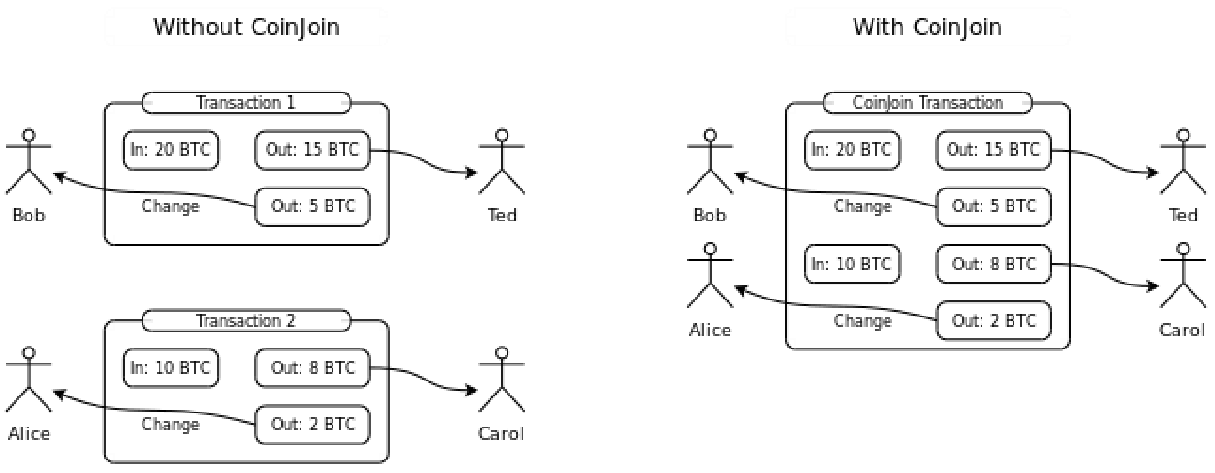


Image 20: Coinjoin [128]

2.5.2.2.2 Coinjoin Security

CoinJoin is actively used in practice, but come with the following set of challenges:

- On the blockchain, some CoinJoins have easily recognizable patterns. Centralized exchanges and other services often halt these transactions for no apparent reason.
- Blockchain analysis services can deanonymize a sender who used CoinJoin, undoing their effort to gain privacy.
- If the number of users required for the transaction is insufficient, the coins can be tracked back via an elimination mechanism.
- CoinJoin is vulnerable to a DoS attack in two ways: someone can refuse to sign a valid joint transaction, or someone can spend their input out of the joint transaction before it finishes. As a result, Coinjoin is vulnerable to misbehaving users.

Since it learns which coins belong to which person, the mixing server must be trusted to maintain anonymity. To address this problem, Maxwell suggests using CoinJoin's safe multi-party computation (SMPC) to conduct the mixing in an oblivious manner. Yang suggests a realistic SMPC sorting system. However, against a completely malicious attacker, both generic and state-of-the-art SMPC sorting are not yet feasible for any fair number of parties required in mixing to ensure a high degree of anonymity. Furthermore, it is unclear if these methods can defend against DoS attacks, since a single user can easily interrupt the entire protocol while remaining undetected. As a result, the Bitcoin community considers defining a practical and safe mixing scheme to be an open issue.

2.5.2.3 Coinshuffle

CoinShuffle, as described in [95], is a fully decentralized protocol that allows users to anonymously mix their coins with those of other interested users, and it is a practical solution for the Bitcoin mixing problem. CoinShuffle is based on CoinJoin, which ensures verifiability, and Dissent, an accountable anonymous community communication protocol that ensures privacy and resistance to active attacks. It includes a blame mechanism in which misbehaving users can be removed from subsequent mixing attempts by the honest partners, ensuring that a joint transaction can ultimately go through.

CoinShuffle does not require the involvement of a third party and is fully compliant with the existing Bitcoin system. It is executed exclusively by the Bitcoin users who want their Bitcoin transactions to be unlinkable. The unlinkability of transactions is guaranteed as long as at least any two participants in a run of the protocol are honest. The protocol uses only common primitives like signatures and public-key encryption, and it is similar to decryption mix networks.

In the absence of a third-party service provider, CoinShuffle does not charge any extra mixing fees to its users. It also performs well in terms of Bitcoin transaction fees, because the participants are only paying for a single mixing transaction.

CoinShuffle has a limited communication overhead for its users, while eliminating additional anonymization fees and reducing computation and communication overhead for the rest of the Bitcoin system.

2.5.2.3.1 How Coinshuffle works

The main part of the CoinShuffle protocol is divided into three phases. An additional blame phase will be reached if the protocol does not run successfully.

Assume that each participant has the same number of coins in their Bitcoin address. This address will be one of the mixing transaction's input addresses, and every protocol message will be signed with the sender's private signing key, which is associated with this address, [95].

- **Announcement**: The participants announce their input addresses.
- **Shuffling**: They perform a shuffling of fresh output addresses. Every participant generates a fresh Bitcoin address, that will be used as her output address in the mixing transaction. Then, like a decryption mix network, the participants shuffle the freshly generated output addresses in an oblivious manner.
- **Transaction Verification**: The participants check the final list of output addresses to see if their address is included. The transaction is signed by the participants and sent to the Bitcoin network in this case (left-hand side). If, on the other hand, an output address is missing (e.g., C' has been replaced by D', right-hand side), the

transaction in invalidated, and the participants enter the blame phase, in which they try to figure out who broke the protocol.

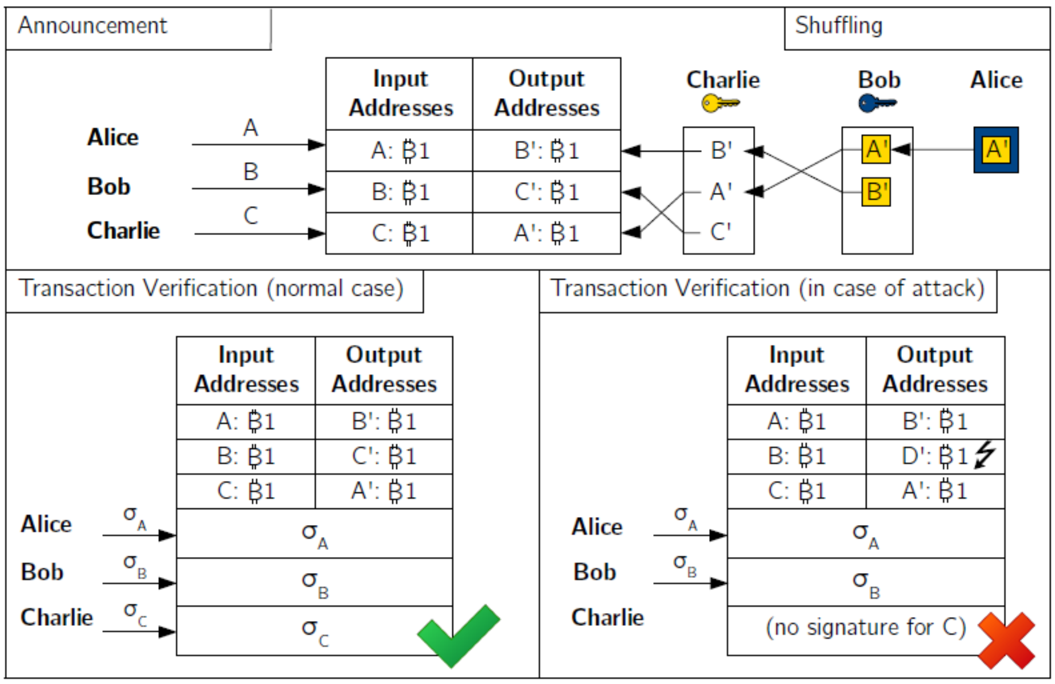


Image 21: Coinshuflle [95]

2.5.2.3.2 Coinsuffle Security

The issues that Coinsuffle has to deal with are as follows, [95]:

- CoinShuffle is also vulnerable to DoS attacks, in which a user enters the mix and then leaves, causing the protocol to be disrupted for all other users.
- In order to identify each other and mix payments, decentralization often requires mix users interacting through a peer-to-peer network. Communication grows quadratically as a result of user coordination, reducing scalability.
- As a consequence of user coordination, communication expands quadratically, decreasing scalability.

2.5.2.4 Xim Protocol

Xim is a two-party mixing protocol compliant with Bitcoin and related virtual currencies, according to [118]. It's the first decentralized protocol to counter Sybil attackers, denial-of-service attacks, and timing-based inference attacks all at the same time. Xim is a multi-round protocol with tunably high success rates. It includes a decentralized system for anonymously finding mix partners using blockchain advertising. No one may confirm or find proof of participants who pair up from the outside. The attacker costs increase linearly with the number of participants in Xim's design, and its probabilistic approach to mixing mitigates Sybil-based denial-of-service attack effects.

Xim protects bitcoins from theft and defends against DoS and Sybil attacks with fees. Users must pay to participate in a mix, raising the bar for attackers that disrupt the protocol by joining the mix and then aborting. Furthermore, an abort by a single Xim user has no effect on the rest of the mix. A tool for finding parties to participate in a mix is one of Xim's core innovations. However, since users must advertise themselves as mix partners on the blockchain, this adds several hours to the protocol.

Xim pools users for mixing using blockchain announcements, eliminating the risk of a single party, such as a bulletin board in P2P mixing or a tumbler, refusing service to honest users and simplifying Sybil attacks by reducing the successful anonymity set of other honest users. However, since Xim only supports two-party mixing, a large number of mixing transactions are needed to achieve even a moderately sized anonymity set.

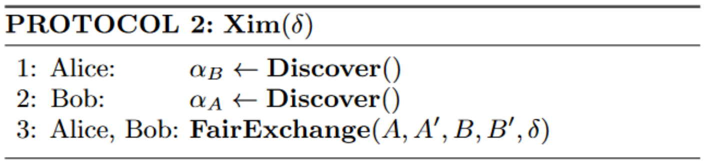
2.5.2.4.1 How Xim works

Xim is described at a high level as follows, [118]:

Alice wants to mix coin stored in address A. To begin, Alice makes a transaction that indicates her willingness to mix with a partner by tipping $\tau/2$ coin from A to the miners; she uses Tor to hide her IP address. She is approached by a number of willing partners (all of whom are using Tor), and she selects one, who then tips τ coin to the miners; no other peers are aware of their partnership. Finally, Alice confirms the partnership by releasing another $\tau/2$ coin to the miners. Once Alice has a partner (let's call him Bob and give him the address B), they exchange funds: δ coin from A are transferred to an address B′ controlled by Bob, while δ coin from B are transferred to A′, controlled by Alice. Using Protocol Fair Exchange, funds are transferred in a single logical step. After this round of mixing is over, Alice starts a new round of ads and looks for a new partner. Alice is expected to have a total of $m\delta$ coin that need to be mixed. As a result, Alice can repeat the protocol m times, but she can do so in parallel if she wishes.

Xim operates in two phases: discovery of a mix partner, and then the fair exchange with that partner.

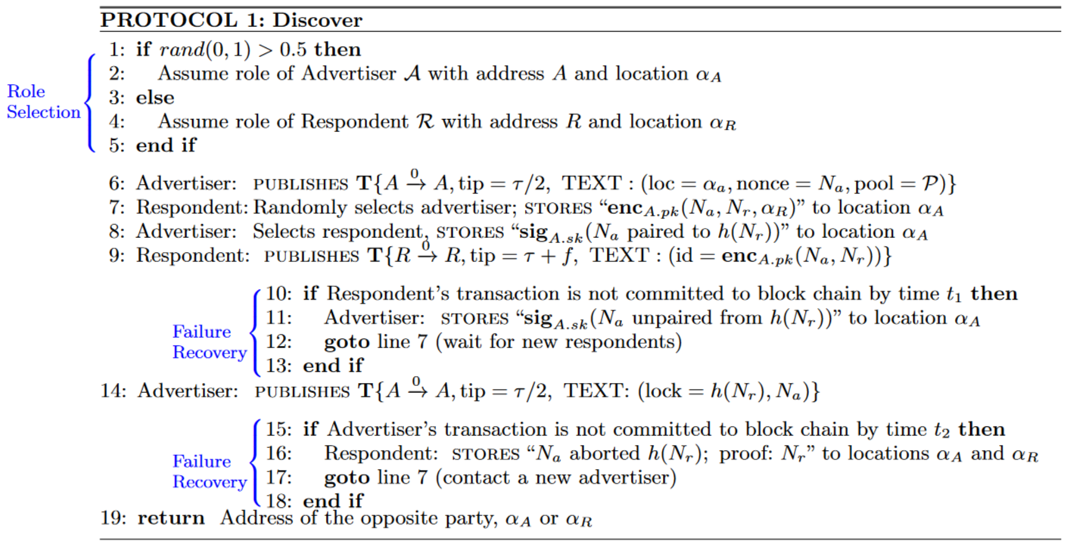
Participants Alice and Bob mix each unit of δ coins over $n \geq 1$ rounds. During any given round, δ coins are transferred from Alice's origin address A to her destination address A′ as follows, and likewise for her partner in the current round, Bob.



```
PROTOCOL 2: Xim($\delta$)
1: Alice:       $\alpha_B \leftarrow$ Discover()
2: Bob:         $\alpha_A \leftarrow$ Discover()
3: Alice, Bob:  FairExchange($A, A', B, B', \delta$)
```

Image 22: Xim [118]

Protocol Discover is responsible for pairing Alice (listening for messages at location α_A) with a randomly chosen participant, Bob, who is listening at location α_B. By design, it is difficult for an adversary to partner herself with Alice in every round.


Role Selection
Failure Recovery
Failure Recovery

PROTOCOL 1: Discover

```
1: if $rand(0,1) > 0.5$ then
2:    Assume role of Advertiser $\mathcal{A}$ with address $A$ and location $\alpha_A$
3: else
4:    Assume role of Respondent $\mathcal{R}$ with address $R$ and location $\alpha_R$
5: end if

6: Advertiser:  PUBLISHES $\mathbf{T}\{A \xrightarrow{0} A, \text{tip} = \tau/2,\ \text{TEXT} : (\text{loc} = \alpha_a, \text{nonce} = N_a, \text{pool} = \mathcal{P})\}$
7: Respondent: Randomly selects advertiser; STORES "$\mathbf{enc}_{A.pk}(N_a, N_r, \alpha_R)$" to location $\alpha_A$
8: Advertiser:  Selects respondent, STORES "$\mathbf{sig}_{A.sk}(N_a$ paired to $h(N_r))$" to location $\alpha_A$
9: Respondent:  PUBLISHES $\mathbf{T}\{R \xrightarrow{0} R, \text{tip} = \tau + f,\ \text{TEXT} : (\text{id} = \mathbf{enc}_{A.pk}(N_a, N_r))\}$

       10: if Respondent's transaction is not committed to block chain by time $t_1$ then
       11:    Advertiser:  STORES "$\mathbf{sig}_{A.sk}(N_a$ unpaired from $h(N_r))$" to location $\alpha_A$
       12:    goto line 7 (wait for new respondents)
       13: end if
14: Advertiser:  PUBLISHES $\mathbf{T}\{A \xrightarrow{0} A, \text{tip} = \tau/2,\ \text{TEXT}: (\text{lock} = h(N_r), N_a)\}$

       15: if Advertiser's transaction is not committed to block chain by time $t_2$ then
       16:    Respondent:  STORES "$N_a$ aborted $h(N_r)$; proof: $N_r$" to locations $\alpha_A$ and $\alpha_R$
       17:    goto line 7 (contact a new advertiser)
       18: end if
19: return  Address of the opposite party, $\alpha_A$ or $\alpha_R$
```

Image 23: Protocol Discover [118]

2.5.2.4.2 Xim Security

Xim is vulnerable to Sybil attacks in various forms. It is possible for one party to present as multiple identities, which can lead a victim to believe they are mixing with an inflated number of parties.
It's vulnerable to intersection attacks, but it has a slight advantage over similar work.
Xim can take advantage of timing attacks. By loosely synchronizing mix participants, these attacks can be mitigated, with the added advantage of reducing other inference attacks by increasing the size of the mix pool, [118].

2.5.2.5 Mobius

Authors in [99] present Mobius, an Ethereum-based tumbler or mixing service. Mobius provides strong notions of anonymity, as even malicious senders are unable to determine which pseudonyms belong to the recipients to whom they sent money, and it can withstand denial-of-service attacks. It also achieves a much lower off-chain communication complexity than any other tumbler, requiring only two initial messages from senders and recipients to participate in an arbitrary number of transactions.

Senders take on the responsibility of transmitting funds to recipients through the tumbler, while recipients take on the responsibility of receiving funds from senders. One of Mobius' key objectives is to reduce the amount of off-chain communication between the sender and the receiver.

The Mobius protocol is designed to scale and currently supports consumer-facing applications for blockchain commerce, and it is substantially less costly than 99.9% of all tokens, [120].

Mobius is a blockchain-based network that aims to mainstream blockchain technology by allowing developers to build their own decentralized apps (DApps) and oracle systems that link data streams and daily applications to the blockchain, [121].

Mobius enables any software developer to become a blockchain developer, providing them with the tools to create a new generation of user-friendly applications that help bridge the gap between blockchain technologies and non-technical users that are unfamiliar with the new technology, [121].

In other words, Mobius is a low-cost mixing system based on smart contracts that offers participants not only strong privacy but also strong availability thanks to the Ethereum blockchain's resilience. Smart contracts remove the need for cooperation between parties wishing to transact in the same mix, confidence in off-chain servers, and the potential for any party (even though all but one colludes) to prevent any honest party's funds from being withdrawn by the intended recipient. The communication cost is minimal, even between parties wishing to transfer funds to each other.

2.5.2.5.1 How Mobius works

Mobius, on the surface, replaces the central tumbler of previous schemes with a mixing Ethereum smart contract, allowing it to operate independently. This gives the system a good sense of availability, as well as anonymity, theft prevention, and low contact overhead, due to the use of stealth keys and ring signatures, [99].

By switching from a central tumbler to a smart contract held on a decentralized blockchain, several issues arose, such as, [99]:

- The ability of a central tumbler to store any hidden information enables the tumbler, and only the tumbler, to release the funds until the rightful recipient attempts to claim them. This is unlikely in a decentralized blockchain since the smart contract's code and all of its storage are also publicly accessible.
- A source of randomness is needed for many cryptographic operations (which are frequently used in tumblers). By its very design, the Ethereum Virtual Machine (EVM) is deterministic: if it weren't, state changes from contracts performed on various computers would be inconsistent, causing nodes to fail to achieve consensus. As a result, we won't be able to use any randomness in the smart contract.

To address these difficulties, the authors leave all secret-key and randomized cryptographic operations to be performed locally by the sender and receiver.

The authors describe how Mobius works, using the following example:

Bob's master public key, a password, and a nonce are shared by Alice and Bob. The ability to generate stealth keys is thus established.

Alice uses the shared secret to generate a fresh stealth public key from the master key any time she wants to give a certain amount of money to Bob.

Even if Alice and Bob never used the addresses involved again, the transaction would create a link between them if Alice sent the money directly to the corresponding stealth address.

To break the link, Alice sends the money and the stealth key to a contract that will mix the exact amount of money. (If Alice wants to submit a sum that isn't protected by a single contract, she must first divide her money into the appropriate denominations.)

The list of stealth public keys it holds is used to form a ring until a sufficient number of senders have paid into the contract. By forming a signature that verifies with respect to the ring in the contract, Bob can now recover his money in an anonymous manner. The simple solution is problematic once again:

Bob could withdraw funds from other recipients in the contract in addition to his own if he forms a regular (unlinkable) ring signature. If Bob withdraws to the same stealth address created by Alice, the link between them will be exposed again (because Alice's transaction exposes the link between her address and his stealth address). To avoid these problems, Bob generates a linkable ring signature to claim the funds (ensuring that he can only withdraw once), as well as a new ephemeral address to obtain the funds (thus ensuring that there is no link between the address that Alice uses to send the funds and the address at which he receives them).

It's important to remember that the latest version of the EVM forbids sending 0-ether transactions to newly created addresses with no value stored in them (which is what Bob does to withdraw from the contract). This will be updated in Metropolis, the next edition of the EVM.

Specifically, according to [99]:

Initializing the contract: The contract, which is kept at address $idcontract$, is set up by specifying the amount of ether it accepts and the number of senders it needs.

This means it's set up with the variables mentioned below.:

• $participants$: number of parties needed to form the ring.

• amt: denomination of ether to be mixed.

• $pubkeys[]$: the public keys over which the ring is formed.

• $senders[]$: the sender addresses.

• $sigs[]$: the signatures seen thus far (used to check for double withdrawal attempts).

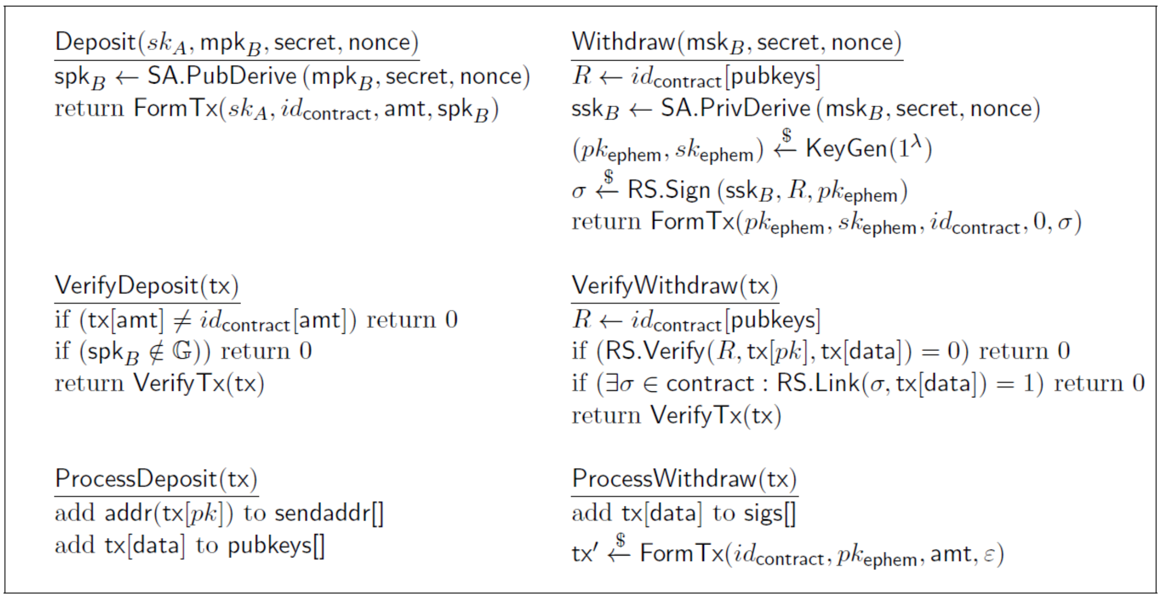
The contract requires code to validate deposit and withdrawal transactions in addition to the code needed to add public keys, addresses, and signatures to the necessary lists.

The contract's storage is removed after the last withdrawal to avoid needing on-chain storage of all signatures used in all previous transactions.

Initializing Alice and Bob: Alice must first be aware of Bob's master public key $mpkB$, before she can send coins to him. They just need to share the password secret and initialize nonce 0 if there is an on-chain public-key directory or if she has prior knowledge of $mpkB$. If such a directory does not exist, the Initialize interaction must also serve to share their master public keys $mpkA$ and $mpkB$, which can be used to share the secret $secret$ using elliptic curve Diffie-Hellman (ECDH) key exchange (or any other mechanism that results in a secret known only to Alice and Bob). This interaction thus enables both Alice and Bob to output $aux = (secret, nonce)$.

Paying in to the contract: Alice creates a new stealth public key for Bob before paying into the deal. She then generates a transaction with amt as the value and the stealth public key as the data and sends it to the contract. The transaction is then broadcasted, and it finally enters the contract. The contract now verifies that this is a legal transaction, meaning that it is properly formatted and includes the correct sum for this contract, as well as that the stealth public key is correct (i.e., has an associated secret key that can be used to withdraw from the contract). It adds the appropriate keys to the lists it maintains if these checks pass. When the requisite number of people have signed up, the smart contract sends a notification to Alice, who processes it. Alice can then run Notify to tell Bob (off-chain) that the contract is ready and sends him the contract address $idcontract$. Alternatively, if the contracts are fixed and have been registered (or Bob otherwise knows $idcontract$), then Bob can process this notification himself and the extra message is not needed. If a fixed time period has passed and not enough people have signed up for the contract, we have two options: either the contract uses $sendaddr[]$ to reimburse the senders who have signed up (which decreases availability), or the contract continues with the mix with the current number of people (which reduces the size of the anonymity set). The contract's maker will decide which of these (or any other) solutions to enforce.

Withdrawing from the contract: Bob retrieves the ring description $pubkeys[]$ from the contract to withdraw from it. He then creates a ring signature with the stealth secret key associated with Alice's stealth public key, allowing him to withdraw his funds from the contract and into an ephemeral address. The contract verifies that this is a legitimate transaction by ensuring that it includes a valid signature for the contract's ring R, that it is not connected to any prior signature used to withdraw from the contract, and that it is correctly established. If these checks are successful, the smart contract saves the signature for future withdrawal verification and generates a transaction sending amt to $addr(pkephem)$. To plan for a new round of mixing, it deletes all stored values except participants and amt once all participants have withdrawn from the deal. There are some possibilities if a certain time period has passed and one or more parties have not withdrawn from the contract. To begin with, one should not set a time limit and instead grant recipients an unspecified period of time to withdraw funds. Second, the maker or the contract may simply return the state to its default state and assert any excess. This decision it is up to the creator of the contract. It is essential that $addr(pkephem)$ is a freshly generated key. If the address corresponding to the public key $spkB$ is used, an eavesdropping adversary might simply hash all the public keys deposited into the ring to see if any match the addresses into which the funds are withdrawn. The sender and receiver addresses will be linked because Alice's deposit transaction contains both her own address $addrA$ and $spkB$. Finally, it's worth noting that if a nonce is ever reused, its stealthiness is lost.



```
Deposit($sk_A$, mpk$_B$, secret, nonce)
    spk$_B$ ← SA.PubDerive (mpk$_B$, secret, nonce)
    return FormTx($sk_A$, $id_{\text{contract}}$, amt, spk$_B$)

Withdraw(msk$_B$, secret, nonce)
    $R \leftarrow id_{\text{contract}}$[pubkeys]
    ssk$_B$ ← SA.PrivDerive (msk$_B$, secret, nonce)
    $(pk_{\text{ephem}}, sk_{\text{ephem}}) \xleftarrow{\$}$ KeyGen($1^\lambda$)
    $\sigma \xleftarrow{\$}$ RS.Sign (ssk$_B$, $R$, $pk_{\text{ephem}}$)
    return FormTx($pk_{\text{ephem}}$, $sk_{\text{ephem}}$, $id_{\text{contract}}$, $0$, $\sigma$)

VerifyDeposit(tx)
    if (tx[amt] $\neq id_{\text{contract}}$[amt]) return 0
    if (spk$_B \notin \mathbb{G}$)) return 0
    return VerifyTx(tx)

VerifyWithdraw(tx)
    $R \leftarrow id_{\text{contract}}$[pubkeys]
    if (RS.Verify($R$, tx[$pk$], tx[data]) $= 0$) return 0
    if ($\exists\sigma \in$ contract : RS.Link($\sigma$, tx[data]) $= 1$) return 0
    return VerifyTx(tx)

ProcessDeposit(tx)
    add addr(tx[$pk$]) to sendaddr[]
    add tx[data] to pubkeys[]

ProcessWithdraw(tx)
    add tx[data] to sigs[]
    tx$' \xleftarrow{\$}$ FormTx($id_{\text{contract}}$, $pk_{\text{ephem}}$, amt, $\varepsilon$)
```

Image 24: Mobius [99]

2.5.2.5.2 Mobius Security

Mobius provides strong protection with an incredibly low attack surface since it was designed from the ground up with a small attack surface.

This makes smart procurement capabilities much more resistant to adversarial assault and auditable, [99].

Mobius satisfies the following:

- Mobius satisfies anonymity, availability, and theft prevention if the stealth address is secure and the ring signature is secure.
- Mobius satisfies recipient anonymity and (partial) sender anonymity with respect to all malicious parties, as well as total anonymity with respect to malicious senders and eavesdroppers, if the stealth address is stealthy and the ring signature is anonymous.
- Mobius satisfies availability if the ring signature satisfies exculpability.
- Mobius satisfies theft prevention if the ring signature satisfies linkability and unforgeability.

2.5.2.6 MixEth

MixEth is presented in [111] and [122]. It's a payment-sending protocol that uses a coin shuffling algorithm to ensure anonymity. The aim is to conceal the identities of individual senders and receivers. MixEth's key breakthrough is its ability to achieve decoy security with low gas costs — costs that are feasible on today's Ethereum.

MixEth is a trustless coin mixing smart contract that allows parties to efficiently tumble coins on Ethereum. It's a Turing-complete blockchain coin mixing operation. MixEth is more powerful than any proposed trustless coin tumbler because it does not require a trusted setup. It only takes 3 on-chain transactions and 1 off-chain message per user. Furthermore, the underlying protocol can be used to shuffle ballots and ciphertexts in a decentralized and trustless manner.

2.5.2.6.1 How MixEth works

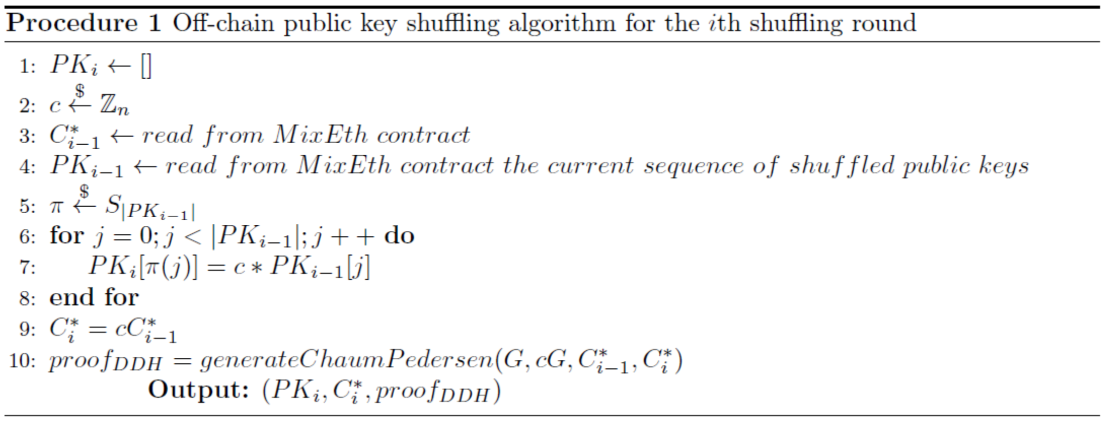
Senders deposit funds and name a public key that can be used to access certain funds in MixEth. Participants — senders, receivers, and everyone else involved — are given the opportunity to perform a verifiable secret shuffle on the public keys until enough deposits have been made to form a large enough anonymity set. Any party may provide evidence that the shuffle was done improperly after each shuffle. The receivers will then withdraw their funds by presenting evidence of ownership of one of the shuffled keys after all interested parties have shuffled.

According to [111], in particular:

Initializing the tumbler and depositing period: The amt parameter, which denotes the denomination of ether to be mixed, must be initialized with a MixEth contract on the Ethereum blockchain at the $id_{contract}$ address. Any sender must deposit the same amount of ether to the same public key. Deposits with an invalid public key or an incorrect ether value are refused. The $initPubKeys[]$ collection is used to store public keys from subsequent deposit transactions.

Shuffling period: Following the depositing round, shuffling and demanding rounds follow in order. Each shuffling round is followed by a challenging round, in which anyone may question the correctness of the previous shuffle. If a challenge is accepted, the shuffler's deposit is lost and returned to the challenger, the incorrect shuffle is discarded, and shuffling resumes with the set of public keys that preceded the discarded shuffle. During a shuffle an honest shuffler can multiply all the public keys by a secret multiplier c and then permute all the transformed public keys.

The new shuffling accumulated constant and the shuffled public keys are sent back to MixEth, along with a Chaum-Pedersen proof showing the validity of the new shuffling accumulated constant. The shuffle is computed off-chain, but the new set of shuffled public keys, the modified shuffling accumulated constant, and the Chaum-Pedersen proof are loaded into the MixEth contract, allowing everyone to check the shuffle's correctness and continue public key shuffling after each challenging round. If the contract is in a shuffling time and the Chaum-Pedersen proof is checked, the mixing contract accepts a shuffling transaction; otherwise, it refuses it.



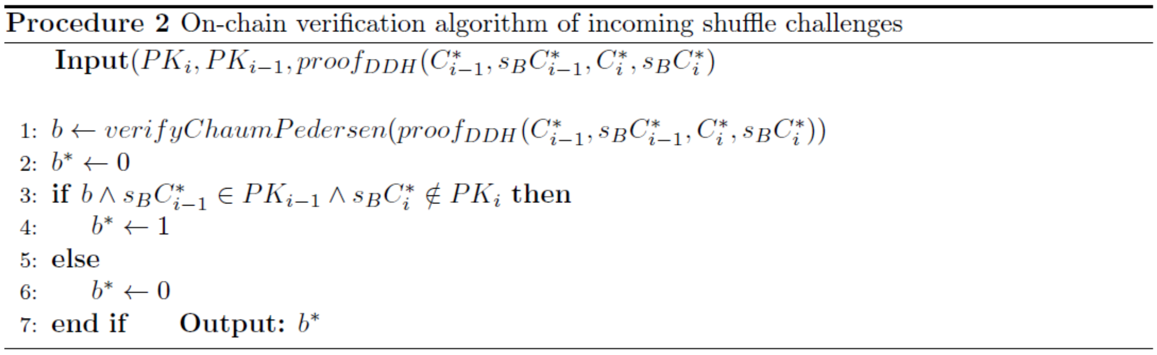
Procedure 1 Off-chain public key shuffling algorithm for the ith shuffling round

```
1:  $PK_i \leftarrow []$
2:  $c \xleftarrow{\$} \mathbb{Z}_n$
3:  $C^*_{i-1} \leftarrow read\ from\ MixEth\ contract$
4:  $PK_{i-1} \leftarrow read\ from\ MixEth\ contract\ the\ current\ sequence\ of\ shuffled\ public\ keys$
5:  $\pi \xleftarrow{\$} S_{|PK_{i-1}|}$
6:  for $j = 0; j < |PK_{i-1}|; j++$ do
7:      $PK_i[\pi(j)] = c * PK_{i-1}[j]$
8:  end for
9:  $C^*_i = cC^*_{i-1}$
10: $proof_{DDH} = generateChaumPedersen(G, cG, C^*_{i-1}, C^*_i)$
        Output: $(PK_i, C^*_i, proof_{DDH})$
```

Image 25: Off-chain public key shuffling algorithm for the ith shuffling round [111]

The function $generateChaumPedersen(G, A, B, C)$ denotes a PPT algorithm, which generates a Chaum-Pedersen proof, proving that $logG(A) = logB(C)$.

Challenging period: Each participant can double-check the accuracy of the incoming shuffles. As a result, each challenging round should be given enough time. If Bob has the secret key s_B, these are the actions Bob as a receiver must take to ensure the shuffle is right at the end of the round. In this case Bob should check whether $s_B\ C^*_i \in PK_i$ or not. If not, Bob must show MixEth that the ith round is, in reality, the first round, and that the shuffled public key corresponding to s_B has been compromised. In the challenge transaction, the Chaum-Pedersen proof guarantees that the legitimacy of the shuffled public key in round i - 1st is preserved, while the integrity of the shuffled public key is compromised in round ith. This verification should be done by each receiver after each shuffling. No one can verify that shuffled public keys for recipients other than themselves are included and right. This role cannot be outsourced unless one exposes her own private key, which will obviously result in a loss of funds at the end of the MixEth protocol, since someone with the corresponding secret key would demand the funds.



Procedure 2 On-chain verification algorithm of incoming shuffle challenges

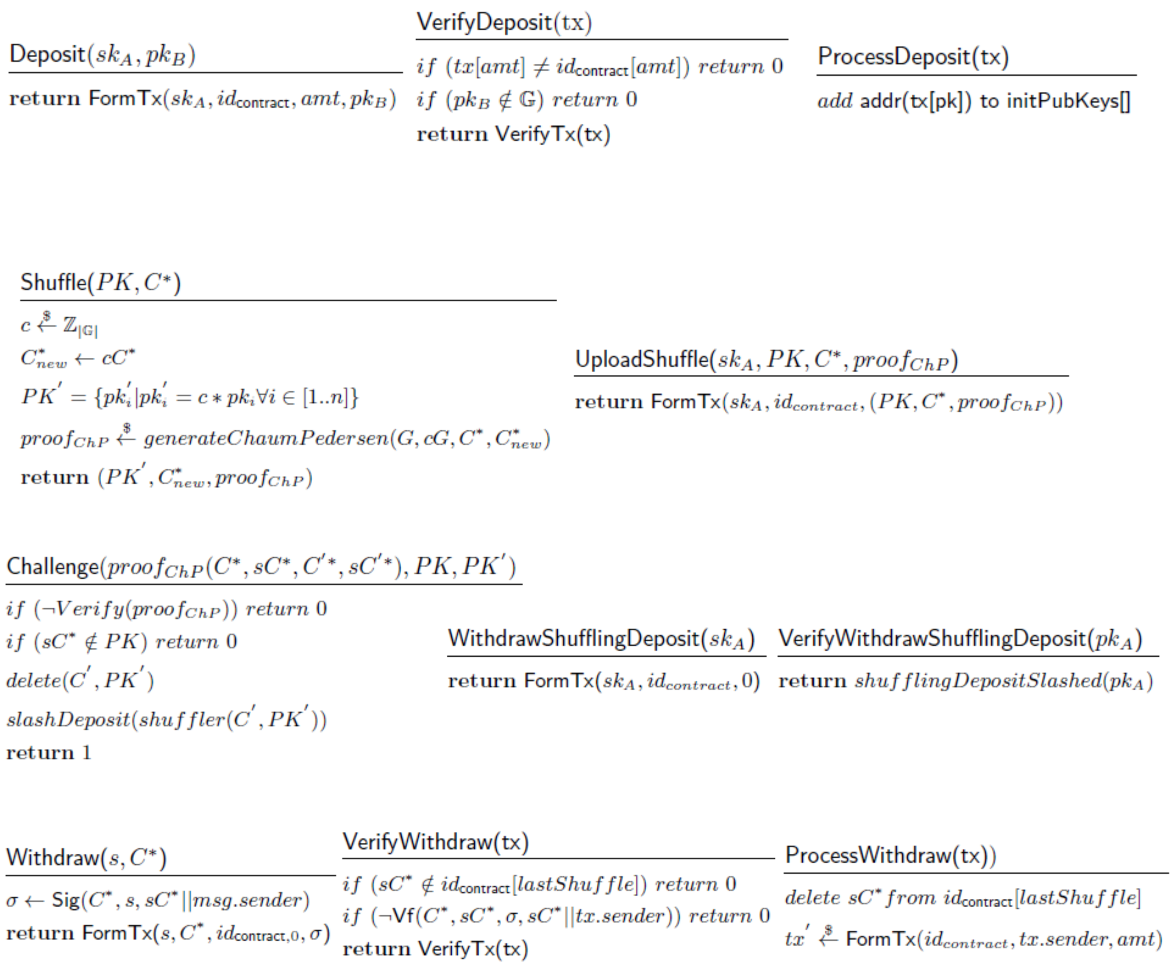
```
    **Input**$(PK_i, PK_{i-1}, proof_{DDH}(C^*_{i-1}, s_B C^*_{i-1}, C^*_i, s_B C^*_i)$

1: $b \leftarrow verifyChaumPedersen(proof_{DDH}(C^*_{i-1}, s_B C^*_{i-1}, C^*_i, s_B C^*_i))$
2: $b^* \leftarrow 0$
3: **if** $b \wedge s_B C^*_{i-1} \in PK_{i-1} \wedge s_B C^*_i \notin PK_i$ **then**
4:     $b^* \leftarrow 1$
5: **else**
6:     $b^* \leftarrow 0$
7: **end if**     **Output:** $b^*$
```

Image 26: On-chain verification algorithm of incoming shuffle challenges [111]

The function $verifyChaumPedersen(proof_{DDH})$ denotes a deterministic polynomial-time algorithm which verifies the correctness of a Chaum-Pedersen zero-knowledge proof. The algorithm outputs 1 if the proof is verified, otherwise 0.

Withdrawing: Let C^*_{final} be the final shuffling accumulated constant. For a recipient B, whose public key $s_B G \in initPubKeys[]$, in the final shuffle there will be $s_B C^*_{final}$. By signing their public key with a changed ECDSA that uses C^*_{final} as the generator element instead of the standardized G, the recipient can prove to MixEth that she knows secret key s_B.



```
Deposit$(sk_A, pk_B)$
    return FormTx$(sk_A, id_{contract}, amt, pk_B)$

VerifyDeposit(tx)
    $if\ (tx[amt] \neq id_{contract}[amt])\ return\ 0$
    $if\ (pk_B \notin \mathbb{G})\ return\ 0$
    return VerifyTx(tx)

ProcessDeposit(tx)
    $add$ addr(tx[pk]) to initPubKeys[]

Shuffle$(PK, C^*)$
    $c \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|}$
    $C^*_{new} \leftarrow cC^*$
    $PK' = \{pk'_i | pk'_i = c * pk_i \forall i \in [1..n]\}$
    $proof_{ChP} \xleftarrow{\$} generateChaumPedersen(G, cG, C^*, C^*_{new})$
    return $(PK', C^*_{new}, proof_{ChP})$

UploadShuffle$(sk_A, PK, C^*, proof_{ChP})$
    return FormTx$(sk_A, id_{contract}, (PK, C^*, proof_{ChP}))$

Challenge$(proof_{ChP}(C^*, sC^*, C'^*, sC'^*), PK, PK')$
    $if\ (\neg Verify(proof_{ChP}))\ return\ 0$
    $if\ (sC^* \notin PK)\ return\ 0$
    $delete(C', PK')$
    $slashDeposit(shuffler(C', PK'))$
    return 1

WithdrawShufflingDeposit$(sk_A)$
    return FormTx$(sk_A, id_{contract}, 0)$

VerifyWithdrawShufflingDeposit$(pk_A)$
    return $shufflingDepositSlashed(pk_A)$

Withdraw$(s, C^*)$
    $\sigma \leftarrow$ Sig$(C^*, s, sC^* || msg.sender)$
    return FormTx$(s, C^*, id_{contract,0}, \sigma)$

VerifyWithdraw(tx)
    $if\ (sC^* \notin id_{contract}[lastShuffle])\ return\ 0$
    $if\ (\neg$Vf$(C^*, sC^*, \sigma, sC^* || tx.sender))\ return\ 0$
    return VerifyTx(tx)

ProcessWithdraw(tx))
    $delete\ sC^*\ from\ id_{contract}[lastShuffle]$
    $tx' \xleftarrow{\$}$ FormTx$(id_{contract}, tx.sender, amt)$
```

Image 27: MixEth [111]

2.5.2.6.2 MixEth Security

MixEth's authors aimed to achieve and prove anonymity, availability, and theft prevention. Specifically, in [111] is referred:

Recipient anonymity: MixEth provides strong anonymity and is resistant to denial-of-service attacks. The funds are sent to the public key $s_B C^*$in the withdrawing transaction for recipient B. if at least one honest sender shuffled and the DDH assumption holds, this public key shows no links to the original $s_B G$. With a small chance of success, the adversary can differentiate between honest recipients' public keys.

Availability: If an adversary is able to deplete the availability of an honest recipient's funds, it means that adversary A has either broken the Chaum-Pedersen protocol's completeness or has successfully launched an eclipse assault against the honest recipient, who is unable to send any transactions to honest Ethereum peers.

Theft prevention: If an adversary were able to steal funds from other people, it means they were able to build a legitimate message/signature,(m, σ) pair for an honest recipient's final shuffled public key without requiring access to the honest recipient's secret key. This challenges the idea that ECDSA is inherently unforgivable.

2.6 Protocol comparison

In this section, there is a comparison between protocols. This comparison is based on [99] and [111].

2.6.1 Security properties achieved by each system

In the following table, the security properties achieved by each system are presented.

Anonymity, availability, and theft prevention definitions have already been disgusted in a previous section (2.4 Security Goal).

In this table, anonymity is classified as protection from three adversaries: an outside eavesdropper, a sender, and a receiver.

In this table, availability is defined as the ability of other participants to cause a sender to waste their time on a transaction that fails, as well as the ability of a tumbler to prevent mixing (which is not applicable in decentralized systems).

Table 5: Security properties achieved by each centralized protocol [99], [111].

	Anonymity against			**Availability**		**Theft prevention**
	Outsiders	**Senders**	**Recipients**	**Sender**	**Tumbler**	
Mixcoin	TTP*	No	Yes	Yes	No	TTP
Blindcoin	Yes	No	Yes	Yes	No	TTP
Tumblebit	Yes	Yes	Yes	Yes	No	Yes

*TTP denotes a Trusted Third Party. In Mixcoin the third party learns the mapping, but under the assumption they do not reveal it, the system achieves anonymity against an outside observer.

Table 6: Security properties achieved by each decentralized protocol [99], [111].

	Anonymity against			**Availability**		**Theft prevention**
	Outsiders	**Senders**	**Recipients**	**Sender**	**Tumbler**	
Coinjoin	Yes	No	Yes	No	n.a	Yes
Coinshuffle	Yes	No	Yes	No	n.a.	Yes
Xim	Yes	No	Yes	Yes	n.a	Yes
Mobius	Yes	Yes	No	Yes	Yes	No
Mixeth	Yes	Yes	No	Yes	Yes	Yes

2.6.2 Proof-of-concept implementation gas cost results

In the following table, it is presented the Proof-of-concept implementation gas cost results between MixEth and Mobius, disgusted in previous sections (2.5.2.5 Mobius, 2.5.2.6 Mixeth).

Table 7: Proof-of-concept implementation gas cost results, [111].

	Deployment	Deposit	Shuffle		Withdraw
			Shuffle upload	Challenge	
Mobius	1,046,027	76,123	0	0	$335{,}714n$
Mixeth	5,395,945	99,254	$366{,}216 + 10{,}000n$	227,563	113,265

2.6.3 Number of on-chain transactions and off-chain messages

The communication complexity needed for each system is shown in the table below in terms of off-chain messages and on-chain transactions. The number of on-chain transactions and off-chain messages are presented per a single participant, required to run a certain coin mixer protocol.

Table 8: Number of on-chain transactions and off-chain messages – Centralized Protocols, [99], [111].

	#off-chain messages	#Transactions
Mixcoin	2	2
Blindcoin	4	2
Tumblebit	12	4

Table 9: Number of on-chain transactions and off-chain messages – Decentralized Protocols, [99], [111].

	#off-chain messages	#Transactions
Coinjoin	$O(n^2)$	1
Coinshuffle	$O(n)$	1
Xim	0	7*
Mobius	2**	2
Mixeth	1	3

* XIM works between two participants, and so cannot be fully compared with the other decentralized mixes, in which n parties can participate.

** In Mobius, the exchange (consisting of two messages) is needed only once per pair, not per transaction, so the cost can be amortized across many transactions.

The cost is captured for a single sender-recipient pair for the centralized ones (because they can mix through the tumbler), while the cost is captured for n participants to mix for the decentralized ones (as they can mix only with each other).

ABBREVIATIONS - ACRONYMS

IOU	I owe you
P2P	Peer-to-peer
CPU	Central Processing Unit
GPU	Graphics Processing Unit
FPGA	Field-Programmable Gate Array
ASIC	Application-Specific Integrated Circuit
PoW	Proof of work
DAG	Directed Acyclic Graph
DNS	Domain Name System
EVM	Ethereum Virtual Machine
ETH	Ethereum
BTC	Bitcoin
PPT	Turing machines that are polynomially-bound and probabilistic are characterized as PPT, which stands for probabilistic polynomial-time machines
NIZKP	Non-Interactive Zero-Knowledge Proof
SMPC	Safe Multi-Party Computation
DApps	Decentralized Applications

REFERENCES

[1] David Shrier, German Canale, Alex Pentland, Mobile Money & Payments: Technology Trends, https://www.getsmarter.com/blog/wp-content/uploads/2017/07/mit_mobile_and_money_payments_report.pdf, 2017.
[2] Prof. Dr. Robby HOUBEN, Alexander SNYERS, Cryptocurrencies and blockchain, https://www.europarl.europa.eu/cmsdata/150761/TAX3%20Study%20on%20cryptocurrencies%20and%20blockchain.pdf , July 2018.
[3] Anonymous, Directed Acyclic Graph, https://en.wikipedia.org/wiki/Directed_acyclic_graph , 2020
[4] Anonymous, Directed Acyclic Graph, https://en.bitcoinwiki.org/wiki/Directed_acyclic_graphs_(DAGs) ,2020
[5] Γεώργιος Ι. Τραχανάς, Ιωάννα Κ. Βρεττού, Κρυπτονομίσματα Τεχνικά χαρακτηριστικά και συγκριτική μελέτη, BSc Thesis, 2019.
[6] Stephanie Bell (Kelton), 'The role of the state and the hierarchy of money', *Cambridge Journal of Economics*, 25, 149-163, http://heteconomist.com/money-interpreted-as-an-iou/, 2015.
[7] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, 2009**.**
[8] Luke Conway, Blockchain Explained, https://www.investopedia.com/terms/b/blockchain.asp, 2020.
[9] Anonymous, Blockchain, https://decryptionary.com/dictionary/blockchain/, 2017.
[10] Anonymous, Blockchain, https://en.wikipedia.org/wiki/Blockchain, 2020.
[11] Anonymous, Minting, https://decryptionary.com/dictionary/minting/, 2017.
[12] Toshendra Kumar Sharma, What Is Solo Mining & How It Works?, https://www.blockchain-council.org/blockchain/solo-mining-works/, 2017.
[13] Anonymous, Pool vs Solo mining, https://en.bitcoin.it/wiki/Pool_vs._solo_mining, 2020.
[14] Anonymous, A brief history on Bitcoin & Cryptocurrencies, https://www.ledger.com/academy/crypto/a-brief-history-on-bitcoin-cryptocurrencies, 2019.
[15] Anonymous, How technology is changing how we value money, https://www.rbcwealthmanagement.com/us/en/research-insights/how-technology-is-changing-how-we-value-money/detail/, 2020.
[16] Jonathan Chiu, The Economics of Cryptocurrencies– Bitcoin and Beyond, https://www.bis.org/events/eopix_1810/chiu_paper.pdf, 2018.
[17] Sergi Delgado-Segura ,Cristina Pérez-Solà , Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas , Joan Borrell, Cryptocurrency Networks: A New P2P Paradigm, https://www.hindawi.com/journals/misy/2018/2159082/, 2018.
[18] Anonymous, Transaction Fees, https://support.blockchain.com/hc/en-us/articles/360000939903-Transaction-fees, 2020.
[19] Finjan Team, Advantages of Cryptocurrency, https://blog.finjan.com/advantages-of-cryptocurrency/, 2018.
[20] David Shrier, Deven Sharma, Alex Pentland, Blockchain & Financial Services: The Fifth Horizon of Networked Innovation , https://www.getsmarter.com/blog/wp-content/uploads/2017/07/mit_blockchain_and_fin_services_report.pdf, 2016.
[21] Jimi S., Blockchain: What are nodes and masternodes?, https://medium.com/coinmonks/blockchain-what-is-a-node-or-masternode-and-what-does-it-do-4d9a4200938f, 2018.
[22] Anonymous, Hash function, https://en.wikipedia.org/wiki/Hash_function, 2020.
[23] Anonymous, Hash function, https://decryptionary.com/dictionary/hash-function/, 2017.
[24] Anonymous, Merkle Tree, https://en.wikipedia.org/wiki/Merkle_tree, 2020.
[25] Jake Frankenfield, Blockchain Wallet, https://www.investopedia.com/terms/b/blockchain-wallet.asp, 2020.

[26] Anonymous, What are blockchain forks?,https://www.cmcmarkets.com/en/learn-cryptocurrencies/what-is-a-blockchain-fork, 2020.

[27] Anonymous, Controlled supply, https://en.bitcoin.it/wiki/Controlled_supply/, 2020.

[28] Anonymous, History of bitcoin, https://en.wikipedia.org/wiki/History_of_bitcoin/ , 2020.

[29] https://coinmarketcap.com/currencies/bitcoin/

[30] Jerome Morrow, What is a Coinbase Transaction, https://blog.cex.io/bitcoin-dictionary/coinbase-transaction-12088 , 2014.

[31] Anonymous, Genesis block, https://en.bitcoin.it/wiki/Genesis_block , 2020.

[32] Anonymous, The Four Planned Stages of the Ethereum Development Roadmap, https://blackwellglobal.com/the-four-planned-stages-of-the-ethereum-development-roadmap/ , 2020.

[33] Anonymous, Ethereum, https://en.wikipedia.org/wiki/Ethereum , 2020.

[34] Anonymous, Bisics Ethereum Phases, https://etherbasics.com/the-basics/phases-of-ethereum/ ,2020**.**

[35] Ryancreatescopy, Transactions, https://ethereum.org/en/developers/docs/transactions/ , 2020.

[36] Rainer Michael, Ripple and the internet of value, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiB_KOV677tAhVBsaQKHTR6DV4QFjAAegQIARAC&url=https%3A%2F%2Fwww.iabfm.org%2Fdownload.php%3Fid%3D431&usg=AOvVaw3j8FdSr0ZrASz5B2ntPbm 7, 2017.

[37] Anonymous, Ripple (payment protocol), https://en.wikipedia.org/wiki/Ripple_(payment_protocol) , 2020.

[38] https://coinmarketcap.com/currencies/xrp/

[39] Steve Florillo, What Is Ripple and How Does it Work, https://www.thestreet.com/investing/what-is-ripple-14644949 , 2018.

[40] Anonymous, Transaction basics, https://xrpl.org/transaction-basics.html , 2020.

[41] Anonymous, XRP Ledger Overview, https://xrpl.org/xrp-ledger-overview.html , 2020.

[42] Andrew Tar, Proof-of-Work, Explained, https://cointelegraph.com/explained/proof-of-work-explained, 2018.

[43] Anonymous, Proof of work, https://en.wikipedia.org/wiki/Proof_of_work, 2020.

[44] Max Thake, what is Proof of Stake? (PoS), https://maxthake.medium.com/what-is-proof-of-stake-pos-479a04581f3a, 2018.

[45] Jake Frankenfield, Proof of Stake (PoS), https://www.investopedia.com/terms/p/proof-stake-pos.asp, 2019.

[46] Anonymous, Proof of stake, https://en.wikipedia.org/wiki/Proof_of_stake, 2020.

[47] Richard Red, Hybrid PoW/PoS Consensus Explained, https://academy.binance.com/en/articles/hybrid-pow-pos-consensus-explained, 2020.

[48] Toshendra Kumar Sharma, A Brief Introduction to Hybrid PoW+PoS Consensus Mechanism, https://www.blockchain-council.org/blockchain/a-brief-introduction-to-hybrid-powpos-consensus-mechanism/, 2020

[49] Kishor Datta Gupta, Abdur Rahman, Subash Poudyal, Mohammad Nurul Huda, M A Parvez Mahmud, A Hybrid POW-POS Implementation Against 51 percent Attack in Cryptocurrency System, https://ieeexplore.ieee.org/document/8968856, 2019.

[50] Anonymous, How Mining Works, https://www.dash.org/mining/ , 2020

[51] Aaron S., DASH Cryptocurrency: Complete Guide, https://www.bitdegree.org/crypto/dash-cryptocurrency, 2020.

[52] Kevin Joey Chen, What is Dash? A step-by-step guide, https://www.finder.com/dash, 2018.

[53] https://coinmarketcap.com/currencies/litecoin/

[54] Anonymous, Litecoin, https://en.wikipedia.org/wiki/Litecoin, 2020.

[55] Anonymous, Litecoin, https://litecoin.org/, 2020.

[56] Jason Fernando, Bitcoin vs. Litecoin: What's the Difference, https://www.investopedia.com/articles/investing/042015/bitcoin-vs-litecoin-whats-difference.asp, 2020.

[57] Anonymous, Difference between Bitcoin and Namecoin, http://www.differencebetween.info/difference-between-bitcoin-and-namecoin ,2020.

[58] Will Kenton, Namecoin, https://www.investopedia.com/terms/n/namecoin.asp, 2020.

[59] Anonymous, Namecoin, https://en.wikipedia.org/wiki/Namecoin, 2020.

[60] Anonymous, Diem, https://www.diem.com/en-us/, 2020.

[61] Anonymous, Diem (digital currency), https://en.wikipedia.org/wiki/Diem_(digital_currency), 2020.

[62] Anonymous, Diem Security on the network, https://www.diem.com/en-us/security/#overview, 2020.

[63] Anonymous, How Do Ethereum Smart Contracts Work?, https://www.coindesk.com/learn/ethereum-101/ethereum-smart-contracts-work/ ,2020.

[64] Fulldecent, Introduction to smart contracts, https://ethereum.org/en/developers/docs/smart-contracts/, 2020.

[65] Pansay, Oracles, https://ethereum.org/en/developers/docs/oracles/ ,2020.

[66] Ryancreatescopy, Ethereum Virtual Machine (EVM), https://ethereum.org/en/developers/docs/evm/, 2020.

[67] Ryancreatescopy, Ethereum Accounts, https://ethereum.org/en/developers/docs/accounts/, 2020.

[68] Ryancreatescopy, Transactions, https://ethereum.org/en/developers/docs/transactions/, 2020.

[69] Baub, Gas and Fees, ,https://ethereum.org/en/developers/docs/gas/, 2020.

[70] Ameer Rosic, Smart Contracts: The Blockchain Technology That Will Replace Lawyers, https://blockgeeks.com/guides/smart-contracts/ , 2020.

[71] Anonymous, Contract, https://en.bitcoin.it/wiki/Contract, 2020.

[72] Anonymous, What is Ethereum?, https://docs.ethhub.io/ethereum-basics/what-is-ethereum/#what-are-smart-contracts-and-decentralized-applications ,2020.

[73] Henrique Moreira, Smart Contracts and Solidity ,https://github.com/ethereumbook/ethereumbook/blob/develop/07smart-contracts-solidity.asciidoc#what-is-a-smart-contract ,2020.

[74] Anonymous, 10 Use Cases of Smart Contracts, https://www.devteam.space/blog/10-uses-for-smart-contracts/ ,2020.

[75] Anonymous, Smart Contract, https://en.wikipedia.org/wiki/Smart_contract , 2020.

[76] Jake Frankenfield, 51% attack, https://www.investopedia.com/terms/1/51-attack.asp , 2019.

[77] https://av.sc.com/corp-en/content/images/blockchain11.png

[78] https://qph.fs.quoracdn.net/main-qimg-d976e3489809ba879a309da6b92e277d

[79] https://sectigostore.com/blog/wp-content/uploads/2019/09/how-hashing-works.png

[80] https://miro.medium.com/max/2880/1*ri4GQEZLa_wYvRsXzMrjow.jpeg

[81] https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/Topological_Ordering.svg/1200px-Topological_Ordering.svg.png

[82] https://miro.medium.com/proxy/1*VMr0S-00QfEeL17wBebhgA.jpeg

[83] https://coinerblog.com/wp-content/uploads/2020/03/Bitcoin-was-Designed-for-a-Financial-Crisis-%E2%80%94-So-far-it%E2%80%99s-Working-well-735x400.png

[84] https://d32myzxfxyl12w.cloudfront.net/assets/images/article_images/d1969cc26a13ed4d400b966e895ef98355817494.jpg?1548936575

[85] https://cdn-images-1.medium.com/max/1600/1*4EQFjXD2-dbiVgVv-8Si8g.png

[86] https://intellipaat.com/mediaFiles/2019/02/Blockchain4-01.jpg

[87] https://neo-ngd.github.io/NEO-Tutorial/en/9-smartContract/imgs/smart-contracts.png

[88] Peter Wind, What is Proof of Stake? PoS Pros and Cons Explained, https://coincodex.com/article/7142/what-is-proof-of-stake/ , 2020.

[89] Aggelos Kiayias, Cryptography Primitives and Protocols, https://crypto.di.uoa.gr/class/Kryptographia/Semeioseis_files/Cryptograph_Primitives_and_Protocols-jun2020.pdf , 2020.

[90] Anonymous, What is a cryptocurrency mixer, https://www.eurospider.com/en/know-how/compliance/211-what-is-a-cryptocurrency-mixer , 2019.

[91] Anonymous, Mixing service, https://en.bitcoinwiki.org/wiki/Mixing_service , 2020.

[92] Dionysis Xenakis, Blockchain-driven mobile data access towards fully decentralized mobile video trading in 5G networks, Access IEEE, 2020.

[93] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll and Edward W. Felten, Mixcoin - Anonymity for Bitcoin with accountable mixes, https://eprint.iacr.org/2014/077 ,2014.

[94] Luke Valenta and Brendan Rowan, Blindcoin - Blinded, Accountable Mixes for Bitcoin, https://fc15.ifca.ai/preproceedings/bitcoin/paper_3.pdf , 2015.

[95] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate, CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin, https://petsymposium.org/2014/papers/Ruffing.pdf , 2015.

[96] Bill Buchanan, Ring Signatures And Anonymisation, https://medium.com/asecuritysite-when-bob-met-alice/ring-signatures-and-anonymisation-c9640f08a193 ,2018.

[97] Brian Curran, What Are Ring Signatures? Providing Privacy for Cryptocurrency, https://blockonomi.com/ring-signatures/ , 2018.

[98] Anonymous, Ring signatues and their applications, http://cryptowiki.net/index.php?title=Ring_signatures_and_their_applications , 2016.

[99] Sarah Meiklejohn and Rebekah Mercer, Mobius: Trustless Tumbling for Transaction Privacy, https://eprint.iacr.org/2017/881.pdf ,2017.

[100] Anonymous, Blind Signature, https://en.wikipedia.org/wiki/Blind_signature , 2021.

[101] John Vacca, Computer and Information Security Handbook, https://www.sciencedirect.com/topics/computer-science/blind-signature ,2017.

[102] Chirag Bhardwaj, What is Zero-Knowledge Proof & its Role in the Blockchain World?, https://appinventiv.com/blog/zero-knowledge-proof-blockchain/ ,2020.

[103] Anonymous, Zero-knowledge proof, https://en.wikipedia.org/wiki/Zero-knowledge_proof/ ,2021.

[104] Anonymous, RSA, https://el.wikipedia.org/wiki/RSA , 2019

[105] Anonymous, RSA (cryptosystem), https://en.wikipedia.org/wiki/RSA_(cryptosystem) , 2021.

[106] Giorgi Mikhelidze, Bitcoin Mixers: Centralized Vs. Decentralized Mixers, https://en.cryptonomist.ch/2020/08/15/bitcoin-mixers-centralized-decentralized/ , 2020.

[107] Anonymous, What Are Bitcoin Mixers?, https://bitcoinmagazine.com/what-is-bitcoin/what-are-bitcoin-mixers , 2020.

[108] Poiresel , Decentralized Mixing, https://www.deltadeltaandmoredeltas.com/decentralized-mixing/ , 2018

[109] Mark, CoinJoin Explained, https://www.mycryptopedia.com/coinjoin-explained/ , 2019

[110] Bruno Škvorc, What is a "bitcoin mixer" and what is a "cryptocurrency tumbler"?, https://audithor.io/bitcoin-mixers/ , 2018.

[111] Istvan Andras Seres, Daniel A. Nagy, Chris Buckland, and Peter Burcsi, MixEth: efficient, trustless coin mixing service for Ethereum, https://eprint.iacr.org/2019/341.pdf , 2019.

[112] Ethan Heilman, Leen AlShenibr, Foteini Baldimtsi, Alessandra Scafuro and Sharon Goldberg, TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub, https://eprint.iacr.org/2016/575.pdf , 2016.

[113] Jordan Clifford, TumbleBit, https://medium.com/scalar-capital/tumblebit-96b346f2e86b, 2019.

[114] Anonymous, CoinJoin, https://en.bitcoin.it/wiki/CoinJoin , 2020.

[115] Adam Hayes, CoinJoin, https://www.investopedia.com/terms/c/coinjoin.asp/ , 2021.

[116] Anonymous, Coin Mixing and CoinJoins Explained, https://academy.binance.com/en/articles/coin-mixing-and-coinjoins-explained , 2021.

[117] Riccardo Masutti, What is CoinJoin?, https://blog.wasabiwallet.io/what-is-a-coinjoin/ , 2020.

[118] George Bissias, A. Pinar Ozisik, Brian N. Levine, Marc Liberatore, Sybil-Resistant Mixing for Bitcoin, https://people.cs.umass.edu/~gbiss/mixing.pdf ,2014.

[119] JP Buntinx , What is the Decentralized Ether Mixer Smart Contract? , https://themerkle.com/what-is-the-decentralized-ether-mixer-smart-contract/ , 2017.

[120] Anonymous, mobius network, https://mobius.network/ , 2021.

[121] William Thrill, Mobius Network Will Help Any Developer Become A Blockchain Developer, https://hackernoon.com/mobius-network-will-help-any-developer-become-a-blockchain-developer-b39fd5f3837f , 2018.

[122]] Chris Buckland, Channelising MixEth with the Counterfactual framework, https://medium.com/anydot/channelising-mixeth-with-the-counterfactual-framework-b53b7f839cc ,2018.

[123] https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.eurospider.com%2Fen%2Fknow-how%2Fcompliance%2F211-what-is-a-cryptocurrency-mixer&psig=AOvVaw05gOvoVbB_GoQhXEPkkMQ0&ust=1617206398298000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCJji0Lqx2O8CFQAAAAAdAAAAABAD

[124] https://i.stack.imgur.com/eCCob.png

[125] https://hackernoon.com/hn-images/1*qlTlvErILjQL9Y7LewAPQA.png

[126] https://lh3.googleusercontent.com/proxy/8DB388xNSnKdMpcAcwfTIJxRnk_TzSNE4lLTqv8whyl7LXlVQuGD3PYckIRaAKYhQIYfW2mnmKBMLOexc76kKkuowk7dWD2wfhEHhM_BE2IDRsXhiISj71A4t0o7XP9ocw

[127] https://upload.wikimedia.org/wikipedia/commons/f/f9/BlindSignature.jpg

[128] https://upload.wikimedia.org/wikipedia/en/thumb/f/f0/CoinJoinExample.svg/640px-CoinJoinExample.svg.png