NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

ENHANCING THE ANONYMITY OF ELECTRONIC TRANSACTIONS

Leonidas A Nasopoulos

Supervisor: **Lazaros Merakos,** NKUA Professor

ATHENS

MAY 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΝΙΣΧΥΣΗ ΤΗΣ ΑΝΩΝΥΜΙΑΣ ΤΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΝΑΛΛΑΓΩΝ

Λεωνίδας Α Νασόπουλος

Επιβλέπων: **Λάζαρος Μεράκος,** Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ
ΜΑΙΟΣ 2021

BSc THESIS

ENHANCING THE ANONYMITY OF ELECTRONIC TRANSACTIONS

Leonidas A Nasopoulos
S.N.: 1115201600115

SUPERVISOR: **Lazaros Merakos,** NKUA Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΝΙΣΧΥΣΗ ΤΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΝΑΛΛΑΓΩΝ

Λεωνίδας Α. Νασόπουλος
Α.Μ.: 1115201600115

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Λάζαρος Μεράκος**, Καθηγητής ΕΚΠΑ

ABSTRACT

Many kinds of online payment systems have been invented during the last decades that allow transactions to be implemented in a more efficient way than the traditional purchases. Also, the online payments do not require physical money. Nevertheless, all such systems utilize a central authority that has the ability to link transactions back to payees and payers.

Since 2009, a new type of independent online monetary system known as cryptocurrency has emerged, permitting clients and recipients to create transactions that are not controlled by a central entity. Such transactions are cryptographically signed transfers of money from client to recipient confirmed by other peers in a global payment network. Due to the fact that confirmation is offered by peers in the network, rather than a central entity, every transaction has to be recorded on a public ledger. This ledger is accessible from every peer inside the network.

To offer some form of anonymity to users in the network, cryptocurrencies like Bitcoin and Ethereum have created their protocols to be pseudo-anonymous. However, this technique only guarantees that a user that generates a transaction cannot be deanonymized if the attacker is observing only one transaction. From a theoretical point of view, since all transactions are visible by peers, attackers can expose the real identities of peers by utilizing other information that is revealed by the network.

In this thesis we perform an in depth analysis of ways to enhance anonymity in cryptocurrencies, and make the de-anonymization of the peers participating in the corresponding network impossible or at least very hard. The main way to achieve this is through mixing services.

SUBJECT AREA: Network Security

KEYWORDS: users, security, cryptocurrencies, anonymity

ΠΕΡΙΛΗΨΗ

Πολλοί διαφορετικοί τύποι διαδικτυακών πληρωμών έχουν αναπτυχτεί τις περασμένες δεκαετίες. Μέσα από αυτά τα συστήματα δίνεται η δυνατότητα στις συναλλαγές να πραγματοποιούνται αποτελεσματικότερα από τις παραδοσιακές συναλλαγές. Επίσης, οι συναλλαγές ολοκληρώνονται χωρίς να απαιτείται η χρήση φυσικού χρήματος. Παρόλα αυτά, όλα τα διαδικτυακά συστήματα πληρωμών χρησιμοποιούν υποχρεωτικά μια κεντρική οντότητα, η οποία έχει την δυνατότητα να αντιστοιχίσει μια συναλλαγή στους χρήστες που συμμετέχουν σε αυτή.

Από το 2009, ένα νέο και καινοτόμο είδος διαδικτυακών πληρωμών σχεδιάστηκε, γνωστό ως κρυπτονόμισμα. Το συγκεκριμένο μοντέλο επέτρεπε στους πελάτες να πραγματοποιούν συναλλαγές με άλλους χρηστές χωρίς να απαιτείται η παρουσία και η χρήση της κεντρικής οντότητας. Αντίθετα με τα πρότερα συστήματα, στα κρυπτονομίσματα οι συναλλαγές υπογράφονται με κρυπτογραφικές τεχνικές και επιβεβαιώνονται από τα υπόλοιπα άτομα του δικτύου. Εξαιτίας του γεγονότος ότι οι συναλλαγές επιβεβαιώνονται από τους χρηστές του δικτύου και όχι από μια κεντρική οντότητα, κάθε συναλλαγή αποθηκεύεται σε ένα δημόσιο πίνακα. Σε αυτόν τον πίνακα έχουν πρόσβαση όλοι οι χρήστες που αποτελούν μέρος του δικτύου.

Για να μπορέσουν τα κρυπτονομίσματα να προσφέρουν μια κάποια μορφή ανωνυμίας, τα σχετικά πρωτόκολλα έχουν σχεδιαστεί με τέτοιο τρόπο ώστε οι χρηστές να αντιπροσωπεύονται από ψευδώνυμα. Όμως η τεχνική αυτή εγγυάται μόνο ότι όταν ένας χρηστής εκκινήσει μια συναλλαγή δεν θα είναι δυνατόν να χάσει την ανωνυμία του, από έναν επιτιθέμενο που παρατηρεί αποκλειστικά αυτή τη συναλλαγή. Σε θεωρητικό επίπεδο, από τη στιγμή που όλες οι συναλλαγές αποθηκεύονται στο δημόσιο πίνακα, οι επιτιθέμενοι μπορούν να παραβιάσουν την ανωνυμία τους εκμεταλλευόμενοι τις υπόλοιπες πληροφορίες που τους παρέχει το δίκτυο.

Η εργασία αυτή αναλύει σε βάθος τρόπους για να ενισχύσουμε την ανωνυμία των χρηστών στα δίκτυα των κρυπτονομισμάτων, έτσι ώστε οι επιτιθέμενοι να μην μπορούν να αντιστοιχίσουν συναλλαγές με χρήστες. Η κύρια τεχνική που εξετάζουμε είναι τα mixing services.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ασφάλεια Δικτύων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: χρήστες, ασφάλεια, κρυπτονομίσματα, ανωνυμία

I dedicate my thesis to my family for the support that they have provided me through all those years

AKNOWLEDGMENTS

I would like to thank Dr. Dionysis Xenakis for his guidance that he provided to me

CONTENTS

LIST OF FIGURES

PREFACE

This thesis is my final project as an undergraduate student in the Department of Informatics and Telecommunication of the National and Kapodistrian University of Athens

1. CRYPTOCCURENCY OVERVIEW

In this introductive chapter we will define the basic cryptographic mechanisms needed in electronic transactions via cryptocurrencies. Also, the basic structure of cryptocurrencies will be explained as well as we will explain every piece of this structure. Furthermore, essential mechanisms that cryptocurrencies utilize will be analyzed in depth. Finally, we will describe the most famous challenges that cryptocurrencies face nowadays as well as the problem statement will be given.

1.1 Cryptographic Tools in Cryptocurrencies

1.1.1 Public Key Cryptography

These kinds of algorithms use a set of keys to produce a secure communication channel among two peers. Each peer owns his unique pair of keys. The first key he owns is the Public key, which is delivered to the other party of the communication. The second key is the Private Key, which is known only by his possessor and it is kept secret from the other party. One very common operation in this kind of cryptography is the Public Key Encryption [1]. In more detail, this technique is implemented when a peer aims to send data to another peer of the network. So, he encrypts the data with the public key of the recipient and sends it through an unsafe communication channel. By the time the recipient receives the data, he uses his secret key in order to decrypt the data. Thus, if the data is stolen from a malicious peer, it is almost impossible to decrypt it without having the correspondant private key. In figure 1 an example of Public Key Encryption is depicted.

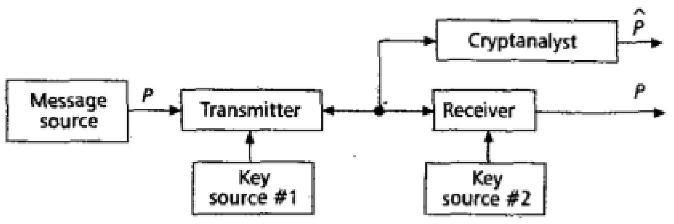


Figure 1: Public Key Cryptography[1]

1.1.2 Secret Sharing

Secret sharing [2] is a cryptographic technique, which divides a secret into a lot of pieces and delivers them to the peers. The secret can be reformed by using the minimum number of parts. The secret's parts are named shares and they are different for every peer. This method is implemented in order to protect classified information.

Shamir's secret sharing is an algorithm that is being implemented in order to deliver transaction data and in the same time the data integrity in the Blockchain will be maintained. This method is also implemented in many off-chain and on-chain cryptocurrency wallets, so the private key of the peers is protected. Let's assume that an organization wants to save its coins with a particular master private key. In this scenario, the secret sharing method would benefit us because we can save the indicated key among a lot of peers. We consider that, three peers will share a cryptocurrency wallet. So we distribute the shares of the key among them. If we examine independently each share we are not able to get any information for the initial private key. But, if the two of the three peers unite their share, they can reform the key. This example is shown in figure 2. This technique can be useful in the Blockchain technology by saving classified information in a decentralized way in order for malicious peers to not be able to observe them.

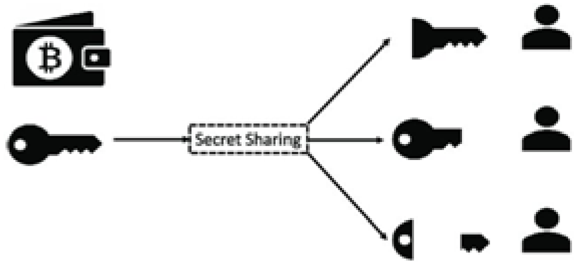


Figure 2: Secret Sharing[8]

1.1.3 Signatures

1.1.3.1 Digital Signatures

The digital signature method is a mathematical idea founded in public-key cryptography, which targets to create short codes, named signatures of digital messages, by the help of a private key. The signatures that are created can only be confirmed from the corresponding public key. The goal of this technique is to prevent the attempts of attackers to tamper and forge digital messages. In the Blockchain technology, digital signatures are implemented in order to sign transactions. The majority of the signatures techniques are highly recommended in order to provide integrity and anonymity in the Blockchain. The signatures scheme is a very essential cryptographic primitive that allows the Blockchain to be publicly confirmed and with achievable consensus. In figure 3, it is depicted the way that a Blockchain peer produces a digitally signed transaction with the help of his private key. Furthermore, figure 4 depicts the way that the other peers of the network confirms, if the signature on the transaction is valid with the help of the public key that belongs to the signer.

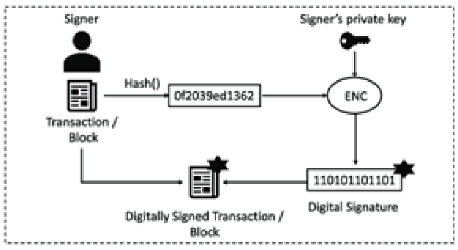


Figure 3: Production of Digitally Signed Transaction[8]

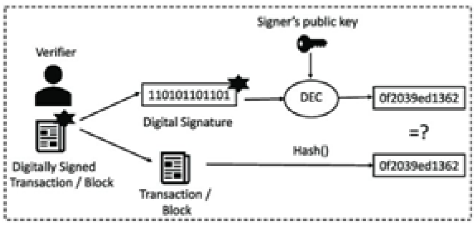


Figure 4:Confirmation of Signature[8]

1.1.3.2 Multi-Signature

In this kind of digital signature, a set of peers must all sign a particular message in order for a process to operate. Inside a Blockchain network, if a transaction demands a signature from a set of peers, it is for our benefit to implement a multi-signature technique. Openchain [3] and MultiChain [4] are Blockchain technologies that use the M-N multi-signature concept. In this way, theft prevention is enhanced because up to M-1 cryptographic keys can be compromised without having any problems.

1.1.3.3 Blind Signature

In the blind signature concept [5], signatures are implemented in privacy-related protocols, in which the peer that signs and the peer that creates the message (transaction in a Blockchain network) are not the same parties. This type of signature is implemented to achieve unlinkability and anonymity of the transaction. The most famous example that uses the blind signature scheme is the Blindcoin mixing network.

1.1.3.4 Ring Signature

The ring signature [6] is implemented by a protocol, where a signature is made by any peer of a group of peers on behalf of the group, while at the same time the identifiers of the signer are not exposed. This technique includes three stages, which are the blinding stage, the signing stage and the unblinding stage. The first one hides the context of the message with a blinding factor. The second one is when the receiver signs the message without knowing its content. Finally, the third one consists of the process to remove the blinded factor and receive the signature.

In other words, it is almost impossible to determine which peer from a given group is the signer. This results in providing anonymity to the signer. The technique of blind signature is implemented by CryptoNote [7] in order to fulfill anonymous payments in the cryptocurrencies.

1.1.3.5 Threshold Signature

The threshold signature idea is a (t, n) signature, where n peers get a share of the private key to produce the signature, and t out of n peers form a signature for any message. During the time that the key is reformed from the peers, by uniting their shares, the key is never exposed publicly. This kind of signature has a great impact on the Blockchain's anonymity. A very famous mixing network which implements threshold signatures is CoinParty.

1.1.4 Zero-Knowledge Proof

In this cryptographic method [9] two entities participate. The first one is the prover and the second is the verifier. Firstly, the prover declares some statements and then he shows their validity to the verifier. The important property is that the prover doesn't expose any personal identifier but only the statements. In conclusion, zero-knowledge proof justifies the statement as 'transfer of an entity is validly' without showing any information connected with the entity. These kinds of cryptographic protocols are very important in order to achieve secrecy in transactions. Zerocoin [10] and Zerocash [11] are two well known systems that their Blockchain's, apply zero-knowledge proofs, so they can accomplish secure and anonymous transactions.

In figure 5, an example of the functionality of the zero knowledge proof is depicted. In the beginning, the verifier is requesting answers in many questions related to the statement. Then, the prover provides at the verifier the answers in a way that it is possible for the statement to be confirmed by the verifier and simultaneously his answers don't expose any additional information

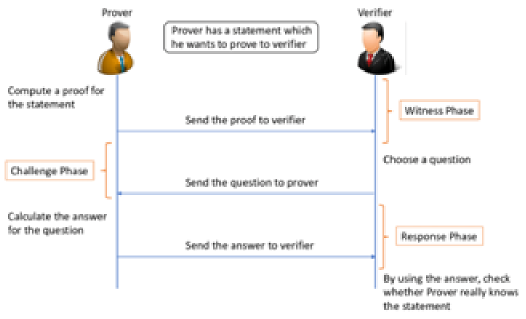

Prover
Verifier
Prover has a statement which he wants to prove to verifier
Compute a proof for the statement
Send the proof to verifier
Witness Phase
Choose a question
Challenge Phase
Send the question to prover
Calculate the answer for the question
Response Phase
Send the answer to verifier
By using the answer, check whether Prover really knows the statement

Figure 5: Zero Knowledge Proof[8]

1.1.5 Hash Functions

These kinds of cryptographic functions [12] receive an input of random size and correspond to them an output with specific size. Hash functions are symbolized with a function H. The most important properties of the hash functions are the following:

1. Collision Resistance: It is almost impossible to think of two inputs, let's say a and b, which will give the same output. In other words, H(a)=H(b) is not possible.
2. Preimage Resistance: For a specific output y it is not easy to find an input that will give H(a)=y
3. Second Preimage Resistance: For a specific input a and output y= H(a) it is not easy to think of an second input b that would result in H(b)=y

The reasons that hash functions are so useful in the Blockchain technologies are the following:

1. They solve complex cryptographic puzzles
2. They generate public addresses (for both public and private keys)
3. They can shorten the public addresses

The most known hash functions that are implemented in the Blockchain are SHA-2 and specifically SHA256 [13].The SHA256 produces outputs of fixed outputs that are 256 bits.

1.2 Transactions

1.2.1 Structure of a Transaction

Concerning transactions in more detail, the coins are depicted with transactions [14] and more accurately, a chain of transactions. As it is shown in figure 6 the main fields of a transaction (we will use as an example here the Bitcoin transactions) are:

- Bitcoin version
- Hash of the transaction
- Locktime
- Inputs (one or more)
- Outputs (one or more)

The inputs include the following fields:

- A Hash pointer to a previous transaction, which is used to identify a transaction that contains the output we desire to use as an input
- A pointer for previous unspent transaction output (UTXO)
- Unlocking script length
- Unlocking script or scriptSig

The outputs include the following fields:

- The amount of coins that are transferred
- Locking script length
- Locking script or scriptPubKey

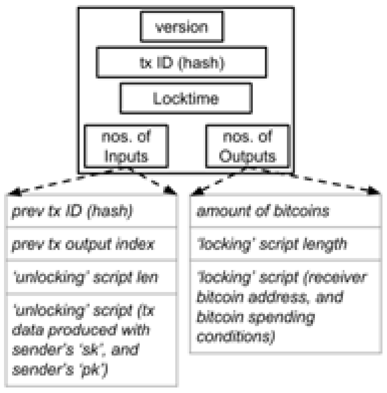


Figure 6: Fields of a Transaction[14]

1.2.2 Flow of a Transaction

A transaction input in order to be licensed, the corresponding peer must provide the public key and the cryptographic signature. Also, in every transaction a list is created that keeps record of the multiple inputs. The value that emerges from adding every input of the transaction is being fully utilized by the transaction's output. Specifically, if the output of the transaction that was implanted before is used as input in a fresh transaction, then the total amount must be spent. There is a possibility that the value of the output is greater than the value of the input. In this scenario the peer forms a fresh Bitcoin address and transfers the difference in this address. An example is given in figure 7

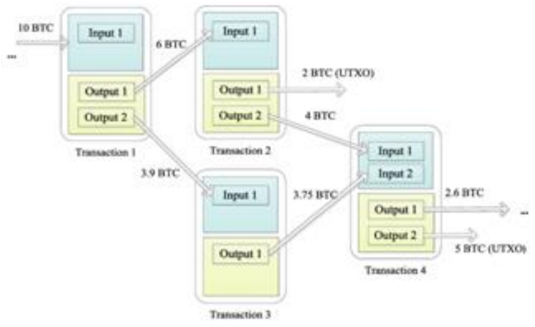


Figure 7: Flow of a Transaction[11]

1.3 Blockchain

1.3.1 Defining the Blockchain

In traditional transactions, central banks confirm, process and save all transactions in a centralized ledger. In contrast with this, in cryptocurrencies, the peers take the role of the bank by maintaining a transcript of the ledger. In cryptocurrencies the role of the ledger is given to the Blockchain [15]. Assuming that a peer in a Blockchain network desires to implement a transaction, his request is saved in a node ledger. This request can be observed by all the peers that are part of the network. The peers' network is responsible to verify the transaction which is stored in the node by achieving consensus. So, by the time the node is authenticated it will be appended in the Blockchain. When the above process is completed, the transaction is impossible to be modified. In case of a malicious peer who wants to attack at the transaction, he must be able to control the whole Blockchain due to the fact that the transaction is accessible by all peers. So, it is considered that to alter a transaction is a scenario that can take place only in theory. Blockchain is a very complex technology due to the fact that it uses mechanisms from different fields of computer science such as cryptography and data structures.

So, the Blockchain is a secure chain of timestamped records as depicted in figure 8, stored in a database that a group of users manage that are part of a decentralized network. The main idea behind the Blockchain is creating a network which consists of multiple peers, who can implement secure and anonymous transactions with each other, without the need of the existence of a third party

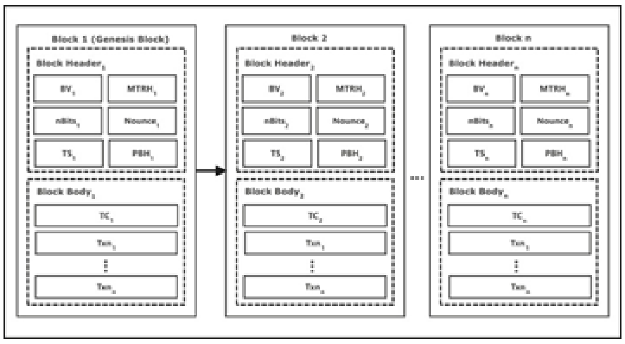


Figure 8:Blockchain[49]

1.3.2 Structure of the Blockchain

The Blockchain is a public ledger (its structure is a linked list) in respect of blocks which keeps records of the network's transaction history. The connection is able to happen due to the fact that every block has a parent block, which is the block before. The initial block in the Blockchain has no parent and it is the genesis block. Every block in the Blockchain includes two parts which are the following:

1. The Block header
2. The Block body

The block headers elements are the following:

- Block version (BV): A network which utilizes the Blockchain technology includes some authentication rules that are necessary to be preserved. So, the version in the block declares the group of rules that must be followed.
- Merkle tree root hash (MTRH) [16]: It is described as the hash value for the whole block. By using the Merkle tree we replace the process of storing the hash value of the entire transaction with a single hash value. This method merges hash values from the whole transaction until a single hash value is created.
- Timestamp (TM):It indicates the contemporary time (in seconds) since 1st January 1970
- nBits: It is the intended threshold of the hash value of an authentic block.
- Nonce: In the beginning it is normally set to zero and every time a hash value is computed the nonce increases
- Parent block hash (PBH): This is a hash value, which size is 256-bit and indicates the previous block.

The block body elements are the following:

- Transaction Counter (TC): It counts the number of the transactions that are included in a block

Transaction: It depicts the transfer of coins or other digital products among two peers. There is the possibility that more than one transaction is kept in a block

1.3.3 Blockchain functionality

An easy way to understand how a Blockchain works [17] is in the below steps:

1. Initially, a peer must request a transaction

2. Then the validation phase takes place. The peers of the network validate the transaction with consensus algorithms which will be described later.
3. By the time the transaction is verified, it will be connected to different transactions to create a new block which will consist of verified transactions
4. After the freshly created block will be added in the Blockchain
5. The process is done.

1.3.4 Taxanomy of Blockchain

1.3.4.1 Public Blockchain

A public Blockchain [18] is permisionless. In other words any peer can have access in the network and read, write or participate in the consensus process. Also, a public Blockchain is decentralized, so it does not exist a central party which is in charge of the network. Every peer in the network is given the same transmission power. Furthermore, information on a public Blockchain are safe because it is impossible to modify them by the time they have been confirmed from the Blockchain. Two famous examples that utilize a public Blockchain, is Bitcoin and Ethereum.

1.3.4.2 Private Blockchain

A public Blockchain [18] is permisionless. In other words any peer can have access in the network and read, write or participate in the consensus process. Also, a public Blockchain is decentralized, so it does not exist a central party which is in charge of the network. Every peer in the network is given the same transmission power. Furthermore, information on a public Blockchain are safe because it is impossible to modify them by the time they have been confirmed from the Blockchain. Two famous examples that utilize a public Blockchain, is Bitcoin and Ethereum.

1.3.4.3 Consortium Blockchain

The consortium Blockchain inherits properties from the public as well as from the private Blockchain. The distinction among the private and the public nature of this type of Blockchain happens on the basis of the consensus process. Here some peers are responsible for the consensus process and the confirmation of transactions. Others peers are assigned to create and review transactions. The distinctions of the roles that peers have can change every time and depends on the consortium Blockchain

The consortium Blockchain may be created in order to inherit more properties of the public Blockchain or the opposite. Depending on which properties are more, it will inherit and its drawbacks.

In figure 9, all types of different Blockchains are depicted

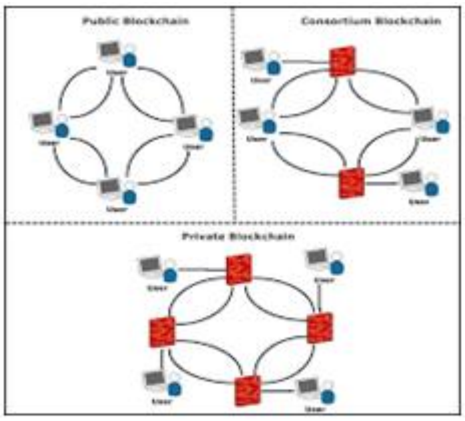


Figure 9: Types of Blockchain[49]

1.3.5 Comparison of the Blockchain Types

1.3.5.1 Throughput

In a public Blockchain, the number of nodes included is high, so it takes a sufficiently large amount of time for propagation, for transactions as well as blocks. Considering the issue of network security, limitations on the public Blockchain are high and this result in increasing the latency. In case of consortium and private Blockchain, due to the fact that there are not many authenticators they are considered to be more efficient.

1.3.5.2 Participation in Consensus Process

In the public Blockchain all peers have the right to participate in the consensus process. In the opposite, the private and the consortium Blockchain, a peer must be authorized in order to be part of the consensus procedure

1.3.5.3 Efficiency

In the public Blockchain, the peers can enter or abandon the network any time, which makes it highly scalable. However, as the complexity of the mining procedure gets greater and the access of new peers to the network gets flexible, which leads to a limited throughput and higher latency. In the other side, due to the fact that the private and the consortium Blockchain consists of fewer validates, they can operate with higher performance and at the same time they consume less energy

1.3.5.4 Central Authority

This is the main variable that makes the Blockchain categories to be distinct. The public Blockchain is characterized from the absence of a central authority, in contrast with the private Blockchain which is fully controlled by a central authority, by the time it is ruled from one entity

1.3.5.5 Transaction Mutability

Blockchain is a decentralized network, so the transactions that participate are stored in varying entities in the computer network. Therefore, it gets almost impossible to modify the public Blockchain. Nevertheless, if there are some influential entities who desire to change the Blockchain, the consortium or private Blockchain can be altered.

1.3.5.6 Read access

In the public Blockchain, due to that it provides access to everybody, this results in making transactions visible to all peers. But, in the case of the private or the consortium Blockchain the authorization for executing read operation depends exclusively on the network.

1.3.5.7 Block Authentication

All peers are part of the process to authenticate a block, in the case of a public Blockchain. Whereas, only limited peers execute the block confirmation in the consortium Blockchain. Finally, in the private Blockchain only one entity is responsible for block authentication

1.3.6 Features of Blockchain

The basic properties that Blockchain technologies have are:

1. Decentralization: In centralized technologies, all transactions demand authentication by a trusted third entity. In Blockchain technology, a trusted third party is not necessary, due to the fact that the consensus protocols can maintain the global view of information consistent in the distributed environment
2. Persistency: In the Blockchain network, the verifications of every transaction are rapidly done. Also, illegitimate transactions will be rejected and not appended in the distributed ledger. Omission or rollback of transactions that are illegal will be identified immediately.
3. Transparent: The information stored in the Blockchain is transparent to every peer. Also every update that takes place in the Blockchain can be publicly viewed by the peers, so Blockchain can be considered as a trusted entity
4. Anonymity: In the Blockchain network, every peer has a produced address by which he is able to communicate with other peers. The produced address does not expose the real identity of the peers that participate in the communication. It is essential to underline that Blockchain does not provide complete privacy due to some inherent constraints.
5. Real-time records: The Blockchain must be renewed as soon as a transaction takes place, with the assistance of some software that will automate the operation. This verifies that every peer keeps its own real time record of its transactions which results in decreasing possible malicious attempts.
6. Open Source: The majority of Blockchain technologies are accessible by everyone. This means that everyone can observe the stored records and also utilize Blockchain in order to produce any application they desire.

1.4 Consensus Process

1.4.1 Defining Consensus Process

As it mentioned above, cryptocurrencies use Blockchain technologies. Blockchain in order to work properly it doesn't only implement cryptographic and P2P technologies but it also uses consensus protocols. Consensus protocols, are methods that make all Blockchain nodes have an agreement in the same message, can make sure the latest block have been added to the chain correctly, guarantee the message that stored by node was the same one. It is essential that the protocols have a balance concerning consistency, availability and partition fault tolerance. Finally, the Byzantine Generals Problem must be faced by the consensus protocols

1.4.2 Types of Consensus Protocols

1.4.2.1 Proof-of-Work

The Proof-of-Work (PoW) consensus protocol is mainly used by the Bitcoin and the Ethereum protocol .This consensus protocol chooses one node in order to generate a fresh block in every round of consensus through computational power competition. The nodes that take part in the competition, must find the solution in a cryptographic puzzle. The participant that manages to first solve the puzzle has the authorization to generate a fresh block. The procedure of solving the cryptographic puzzle is considered difficult, because the nodes have to modify the value of the nonce until they obtain the correct solution. The modification demands a large quantity of computational power. It is possible for a malicious peer to overturn one block in the Blockchain. But, as the chain increases in length the malicious peer's target is getting very hard due to the huge quantity of computational power that is needed. The protocol is depicted in figure 10.

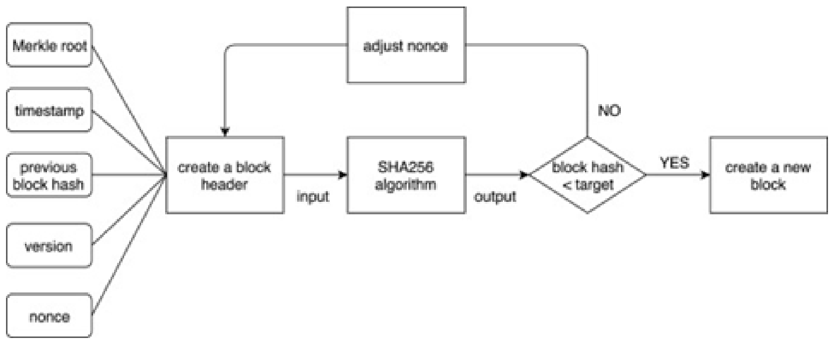


Figure 10: PoW Protocol of Bitcoin[48]

1.4.2.2 Proof-of-Stake

In the Proof of Stake (PoS) algorithm, in every round a peer is selected in order to produce a fresh block. The selection is done according to the proportion of coins the peer possesses, and not his computational power, despite the fact that peers still have to find the solution to a SHA256 puzzle. The main difference from PoW is that peers are not forced to change the value of the nonce consistently. Instead, the main factor to solve the SHA256 puzzle is the amount of stake (coins) they own. PPcoin, was the first digital currency that implemented the PoS protocol to the Blockchain. In this cryptocurrency except the amount of coins, the coin age has a key role in finding the solution to a PoS puzzle [19]. Coin age is defined as currency amount times holding period. Let’s assume that Bob gets 20 coins from Alice and possesses them for 80 days. Then Bob has piled 1600 coin-days of coin age. By the time, a fresh block is created by a peer his age will be initialized to zero. Other cryptocurrencies that adopt PoS are Nxt [20] and Ouroboros [21]. The protocol is depicted in figure 11.

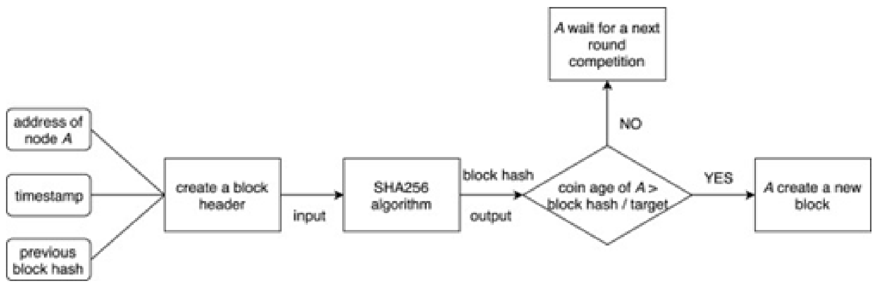


Figure 11:PoS Protocol of PPcoin[48]

1.4.2.3 Delegated-of-Proof of Stake

The Delegated Proof of Stake (DPoS) protocol permits to the nodes who posses stake to vote in order to elect block verifiers [22] (block generators).This method of voting permits the peers that hold the stake to transfer the responsibility of generating new blocks to the block verifiers they vote. This has an important impact in the block creation procedure because the computational power that is now required is zero. In the case that the elected block verifiers cannot produce the block, then the stakeholders will elect new block verifiers in order to replace them. The most known cryptocurrencies implementing this consensus protocol are BitShares and EOS. The protocol is depicted in figure 12

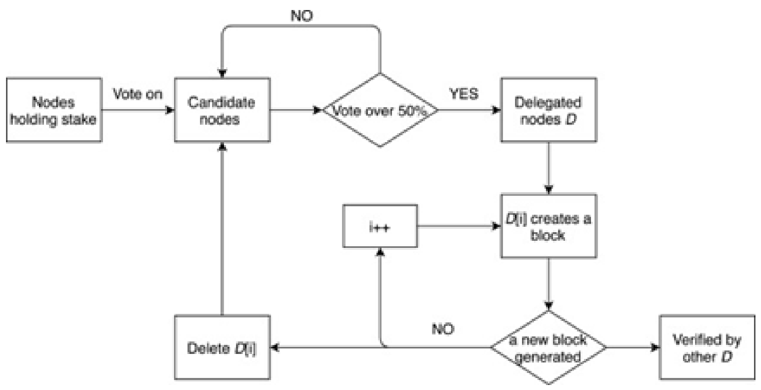


Figure 12: DPoS Protocol of BitShares[48]

1.4.2.4 Ripple

Ripple protocol consists of an open source payment. Transactions are triggered from clients and they are published inside the network by tracking or validating peers. But, the consensus procedure is implemented by validating nodes .Every validator has a list of trusted nodes called Unique Node List (UNL).The nodes that belong in the UNL have the right to vote on the transactions they desire. All validating nodes form a group of transactions that they support and send them to other validators as a proposal. By the time a validator receives a proposal, he will compare every transaction with the transactions that he has in his own proposal. Every transaction that is found in both proposals will get a vote. When a transaction receives more than 50% of the votes, the transaction will be involved in the next round. As the rounds pass the threshold of the votes will increase. The transactions that will finally receive more than 80% of the votes will be appended in the Blockchain. The protocol is depicted in figure 13.

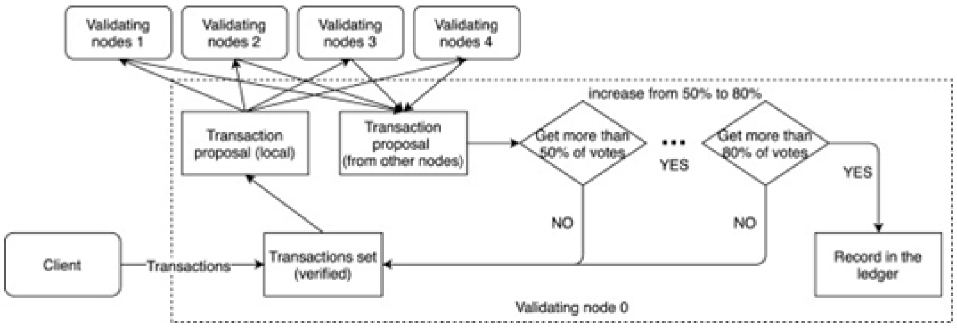


Figure 13: Ripple Protocol[48]

1.4.2.5 Practical Byzantine Fault Tolerance

The Practical Byzantine Fault Tolerance (PBFT) consensus protocol has small complexity and great practicality in distributed networks [23]. The nodes in this protocol are sequentially ordered with one node being the primary (or the leader node) and the others referred to as secondary (or the backup nodes).Note here that any eligible node in the network can became the primary by transitioning from secondary to primary (typically, in the case of a primary node failure).The target is that all honest nodes help in reaching a consensus regarding the state of the system using the majority rule. A practical Byzantine Fault Tolerant system can function on the condition that the maximum number of malicious nodes must not be greater than or equal to one-third of all nodes in the system. As the number of nodes increase the system becomes more secure. The PBFT consists of five stages which are the request, pre-prepare, prepare, commit and reply. In the beginning, the client sends a request to the primary node. Then the primary node broadcasts the request to the 3 secondary (backup) nodes. After the nodes (primary and secondaries) perform the service requested and then they send back a reply to the client. The request is served successfully when the client receives m+1 replies from different nodes in the network with the same result. The m variable represents the maximum number of faulty nodes allowed. The protocol is depicted in figure 14.

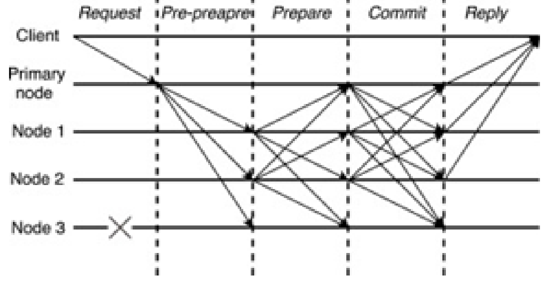


Figure 14: PBFT Protocol[48]

1.5 Mining

1.5.1 Defining the Mining Process

All cryptocurrencies networks as mentioned above, utilize the Blockchain technology. A transaction is generated when a sender transfers coins to a recipient. The mining process is responsible for confirming and appending transactions in the Blockchain. During the execution of a transaction the miners check, if the coins are owned by the sender or he is attempting to double spend. The owner of the coins is information that can be found in the Blockchain. There is the scenario where a malicious peer tries to produce a great number of nodes in order to confirm an invalid transaction. The miners in order to face this case are requested to find the solution to a resource intensive problem. This task is usually, a proof method named as mentioned above consensus protocol. So the consensus protocols convert the process of making a lot of fake nodes, very costly for the malicious peers.

The creation of proof demands intensive utilization of memory and computational power. This technique also limits the transactions that can be confirmed, and as a result of this, it limits the number of blocks that can be appended to the Blockchain in particular time. This limitation is necessary, due to the fact that when a block is mined, fresh coins are produced. So it is important to decrease the rate of production to prohibit untimely exhaustion. Let's assume as an example the Bitcoin, the most famous cryptocurrency. The mining [24] process is completed after the following steps have been executed.

1. A miner executes a resource-intensive task and generates evidence that the task has been completed. This task prohibits a malicious miner from generating fake identities and manipulating figure
2. The evidence generated is verified in order to confirm that the task has been completed
3. The miner after, checks for the verification of the transactions as well as if every transaction inside the block prove verified
4. The block is appended in the Blockchain

Mining belongs to the brute-force algorithms and must be created in order that the number of blocks that are mined per second is permanently stable. In this way, it is possible to control the speed of entering new coins. A reward is given to the miner that will first provide the proof in order to confirm the block. This reward is a fraction of the new coins that will be generated

1.6 Forks

1.6.1 Defining the Fork Problem

Forks depict changes to cryptocurrencies protocol that converts the previous rules to be valid or invalid. In other words, cryptocurrency forks are protocol upgrades. So, by the time an updated version of Blockchain software is released the consensus rules are also renewed. As a result, the existing nodes in the network are distinct in New Nodes and Old Nodes. So, 4 possible scenarios exist

1. The New Nodes agree with the transaction of block which is sending by the old nodes

2. The new nodes don't agree with the transaction of block which is sending by the old nodes.
3. The old nodes agree with the transaction of block which is sending by the new nodes.
4. The old nodes don't agree with the transaction of block which is sending by the new nodes.

Due to these four different scenarios in reaching consensus, fork problems happen, and can be distinct into two types, the Hard Fork and the Soft Fork.

1.6.2 Hard Forks

Hard Forks emerge when the network is upgraded to a new version or functions based on a different agreement. The old version cannot align with the new, which leads the old nodes to not agree with mining of the new nodes. This results in having two chains instead of one. Despite the fact that old nodes have smaller computational power than the now nodes, old will maintain their chain.

By the time a Hard fork occurs, all nodes are asked to upgrade their version. After that nodes which have not completed the request will not function as before. If the majority of nodes did not renew their agreement, then the chain will fork into two different chains because the old nodes will keep forking a different chain.

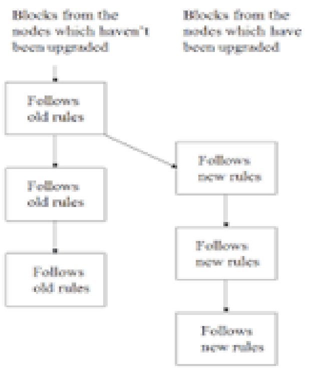


Figure 15: Hard Fork[17]

1.6.3 Soft forks

Soft fork defines the situation where the agreement or version changes and it is not function based on the previous. So, the old and the new nodes are not able to agree with the mining process. Due to the fact that new nodes have a greater computing power in comparison with old nodes, the mining implemented by the old nodes will never be confirmed by the new nodes. But, both types of nodes will still fork the same chain. The nodes in the system are not obligated to upgrade the agreement simultaneously, when soft fork takes place. Instead, it allows the nodes to renew the version gradually. In contrast with Hard Forks, Soft forks will have one chain to fork.

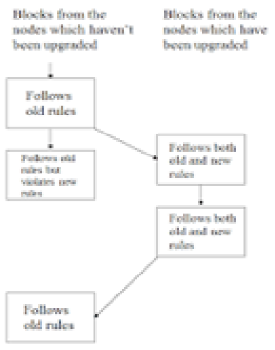

Blocks from the nodes which haven't been upgraded
Blocks from the nodes which have been upgraded
Follows old rules
Follows old rules but violates new rules
Follows both old and new rules
Follows both old and new rules
Follows old rules

Figure 16: Soft Fork[17]

1.7 Smart Contract

A smart contract [25] represents an agreement between disbelieving parties, which is executed by the consensus protocols that are used by the Blockchain. The code and the data that are included in the smart contract are named as methods and states. The anticipated transactions that are obtained by the Blockchain call the smart's contract methods utilizing its data, in order to perform the requested service. Due to the fact that the computer code is included in the Blockchain, it is immutable and may be considered as a third entity to complete complicated financial transactions. The only limitation on the operation of smart contracts is their complexity: every action they execute consumes a specific quantity of gas, which is a subcurrency within Ethereum created to limit the amount of computation that an specific smart contract can utilize. These kinds of contracts are capable of doing calculations as well as recording data for economic transactions.

During the mining phase, miners not only mine blocks but simultaneously they run smart contract code. So, the execution of the code results in increasing the cost, in comparison to transferring coins in other Blockchain based cryptocurrencies. Except of paying the required transaction fees, the sender who is requesting for the transfer to a smart contract, is also charged with fees for code execution. Smart contracts must be executed in a specific time period. If the contract is not executed by that time the transaction is rejected. A smart contract is capable of requesting other contracts to run through messages

1.8 Peer-to-Peer Network

A peer-to-peer network uses unencrypted TCP connections as its basic communication format. The nodes of the network have a list of IP addresses that belong to peers that may be connected in the future. This list is bootstrapped through a DNS server. Every peer has a goal, to store at least 8 decrypted TCP connections. In the case that a peer has less than 8, he attempts to establish new connections. It is defined by the network that peers are waiting in port 8333 for inbound connections.

By the time, peers have created new connections, they implement an application layer handshake which includes the version and the verack messages. The verack message consists of a timestamp for synchronization, IP addresses and the version of the protocol. Every node picks its peers randomly and after a predefined time period, it picks a new group of peers. Finally, in order for the network to be able to keep track of the peers that are leaving, it utilizes a soft state approach. Every 30 minutes the peers will publish a hello message in order to maintain the connection active.

1.9 Double Spending

The most important problem concerning cryptocurrencies is the double spending. Double spending is the attempt of a client to implement two different transactions with different recipients using the same coins. Before cryptocurrencies were invented, the issue was faced with a central entity that was responsible for verifying transactions. Instead, cryptocurrencies introduced the Blockchain, which is responsible for the approval of a transaction. If a peer attempts to spend the same amount of coins twice, the transaction is rejected by the Blockchain. The Blockchain has this ability due to the consensus rules. These rules are alternatively called validation rules because transactions and blocks are confirmed based on them

1.10 Cryptocurrencies

1.10.1 Bitcoin

Bitcoin [26] is without doubt the most famous and dominant cryptocurrency. It was the initial realization of the concept of a new kind of money that utilizes cryptography in order to manage transactions, without a central entity. Bitcoin has a decentralized nature which means that the network is controlled and owned from all peers. Also, it means that every peer must obey the same rules. Furthermore, Bitcoin utilizes Blockchain technology, which stores the history of all transactions. In addition, due to the Blockchain, the processing and verification are implemented by the peers participating in the network .Despite the fact of the decentralized nature which offers a great number of advantages some people criticize the absence of an entity that would overlook the whole network. In return of offering their computing resources to Bitcoin in order to complete the mining process, the peers are being rewarded with coins.

1.10.2 Ethereum

In the Ethereum (ETH) [27] network, the Blockchain technology targets to offer smart contracts, which in reality are programming codes that are placed on the Blockchain and can be reached by the peers of ETH. The smart contract has the ability to receive and transfer coins and simultaneously it can implement complex calculations. If the contract is correctly created then it can behave as a trusted third party, in case of economic transactions, due to the fact that the code is public and also immutable. The platform of ETH utilizes a Turing complete language for transaction programming. Furthermore, coins and transactions fees are given to the miner. Gas consist the fuel to the transactional calculations. Every transaction utilizes gas while it takes place, and the creator of a particular transaction must provide sufficient gas. If this gas is not adequate, than the transaction will stop executing

1.10.3 Litecoin

The Litecoin [28] cryptocurrency was produced in 2011 by Charles Lee with the assistance of the Bitcoin network. Founded on the exact same peer-to-peer protocol that is applied by Bitcoin, it is often characterized as Bitcoins main rival. The Litecoin uses the proof-of-work consensus algorithm and it has a significantly smaller confirmation time for transactions.

1.10.4 Dash

Dash [29] is also known as DarCoin or XCoin and consists of a privacy-centric digital currency with instant transactions. Despite the fact that it is founded on Bitcoin and it has similar elements, Dash is a two-twired network. Dash is supervised from a set of servers which are called Masternodes, which eliminates the necessity of a third party. This enhances the ability to implement transactions, which are fast and with an improved privacy. On the other side, nodes or miners inside the network offer the computing power for basic operations like defending against double spending attempts as well as transferring and obtaining coins. The primary advantage, in comparison with the Bitcoin network is that the transactions can be verified almost in real time. This happens because the Masternodes are not the same peers with the miners. The Dash network uses the X11 chained PoW hashing algorithm, which aids to dispense the processing equally across the network, while keeping a near coin distribution to Bitcoin. Dash is able to decrease security flaws because it utilizes eleven different hashes.

1.10.5 Monero

Monero (XMR) [30] currency is considered to be secure, private as well as untraceable. The currency is totally donation-based, community driven and founded on the PoW consensus algorithm. Despite the fact that transactions inside the network are private, peers can adjust their level of privacy permitting access in the transactions they own at the scale they desire. Despite the fact that it implements a PoW consensus algorithm, Monero have many similarities with Litecoin concerning the mining procedure of the coin. This happens because the mining can be implemented by all synchronous computers and it is not limited to particular created hardware. This cryptocurrency has some advantages in comparison with the Bitcoin, such as it utilizes a dynamic block size, so it can solve the scalability issue.

1.10.6 Ripple

Ripple (XRP) [31] is a digital currency that targets to build on the Bitcoin technique and also aims to connect various payment systems to one another. It was initially created in 2012.This cryptocurrency is global real-time gross settlement network (RTGS) which allows banks to transfer real-time international payments across networks. This network is a Blockchain based system.

1.11 Challenges

1.11.1 Scalability

The recent increase of the utilization of the cryptocurrencies has negatively affected its ability to scale [32] with a large number of users. It is not surprising that a network with only one entity that stores transactions, the Blockchain, can display a bottleneck. The scalability problem can be depicted in different ways .The first one is the latency which is the period that is needed for the confirmation of the transaction. The second one is the bootstrap which is the duration a new node requires in order to obtain and process the history that is needed to validate payments. As we can observe there are a variety of ways to evaluate a digital cash system. The most comprehensive seems to be the throughput, which has a direct impact on scalability. The throughput can be thought as the variable that counts the maximum number of peers a network can handle per second. Throughput in the Bitcoin network can be evaluated through the network communication latency to the processing power of the entities that evaluate the transactions. However, the most important evaluation measure is the maximum size of blocks.

1.11.2 High Energy Consumption

Bitcoin Blockchain utilizes the PoW idea to accomplish distributed consensus in the network. PoW forces the mining process to be more resistant to various security breaches like Sybil attacks and double spending, but it also consumes a large quantity of energy and computing resources [33]. Therefore innovative technologies that decrease the energy consumption are needed in order to assure a better future for Bitcoin. Finally, the consistently increasing network load and energy consumption increases the duration that is needed for a transaction to get processed

1.11.3 Wallets can be lost

By the time cryptocurrencies do not implement transactions using a third party, there is a possibility for a peer to lose his private key that is linked to his wallet, due to the fact of a hard drive crash or even a virus that can corrupt data. In such a case, all the coins placed in the wallet will be considered as lost. The Bitcoin network will not have any way to help the peer in regaining the coins.

1.11.4 Anonymity

A great issue in all cryptocurrencies is maintaining the anonymity of the peers in a transaction. This means given a specific transaction, an outsider must not be able to distinguish its sender and recipient. Also due to the fact that the transactions are visible by all peers it is possible that the identity of peers can be exposed. For example some possible attempts to deanonymize the peer id to link transactions to IP addresses or third party applications can be used in order to link the peers different profiles and currencies. So, due to a lot of attacks that have been invented concerning the observation of the Blockchain and not only, it is considered a great challenge to enhance the anonymity in digital cash systems

1.11.5 Throughput

Throughput is variable that is evaluated according to the number of blocks that are added in the Blockchain per second. In other words it represents the number of transactions evaluated per second. Some factors that affect the efficiency of throughput are the number of peers that take place in the consensus process, the network infrastructure, the block parameters as well as the complexity smart contract(in the scenario of smart contract supported networks).So considering the above factors it is easily understood that maintaining a high throughput is a complex task

1.12 Anonymity Properties

1.12.1 Defining Pseudonymity

Pseudonymity is an essential property that was already offered in peers from cryptocurrencies from the first versions of Bitcoin. This property just helps to record transactions as transfers of coins among one public key owned by the client to another public key owned by the recipient, instead of between the real world identities of communicating end points. So pseudonymity just breaks the connection among a public and a real world identity. Once more from a practical point of view, if it is impossible for an attacker to recreate the connection among the public key and the real world identity, than it is assumed that the pseudonymity is effective for maintaining the peer's anonymity.

1.12.2 Defining Unlikability

Unlikability defines that the user participates in transactions inside the network repeatedly. The different transactions that have the participation of the peer must not be possible to be tied up to one another, from the aspect of a malicious peer. A transaction can be said that is unlikable, if it impossible to connect different addresses or transactions of a receiver as well as of a sender and when nobody can correspond a client's transfer to a recipient. Unlikability in a more formal way is defined as the property that for two outgoing transactions with recipients A and B, it must be impossible or computational very expensive, to provide evidence that they were received from the same peer (A=B).In the same way, for any two incoming transactions with clients A and B it must be impossible or computational very expensive, to show that they were triggered by the same peer (A=B).

The property of unlikability can be also implemented to a single transaction. In more details it is meant that the transaction cannot be traceable back to a particular client or recipient. This kind of unlikability is named as untraceability. Once more in order to describe it more formally, it is defined that given a specific transaction input, the output that will occur must be anonymous between a group of different outputs.

1.12.3 Defining Anonymity

Before we can claim that a cryptocurrency protocol provides anonymity to its peers, it is essential to first comprehend what is defined with deanonymization concerning the transactions. From a practical point of view, when a attacker achieves to identify the real world address of the client, the recipient as well as the number of coins that are being sent, than we have a successful deanonymization attempt. By the time the Blockchain is visible by all peers, a transaction will be deanonymized if the real addresses of the possessors of these public keys were by choice exposed on online forums. This will take place due to the fact that it would then be trivial for an attacker to connect these keys to real addresses of the peers.

Nonetheless, from a theoretical point of view, we take into consideration the quantity of information that an attacker already has knowledge of. It can be supported that anonymity is violated when the network has a weakness that the attacker can take advantage of in order to obtain extra information and increase the chances to expose the identity of peers. For instance, anonymity is violated when a transaction can be connected to a specific sender or recipient because this gives more probabilities to the attacker to identify all entities. So, researchers have come to a conclusion that anonymity in cryptocurrencies is defined as: anonymity = pseudonymity + unlinkability

1.13 Deanonymization of Peers

A malicious peer, who desires to deanonymize a transaction, must form a one-to-many mapping among the peer and his associated addresses. Specifically a peer can be connected to a group of addresses by utilizing a Blockchain analysis process [34].The above analysis demands 3 pre-proceeding phases, which are:

1. Transaction graph: The entire ledger can be thought as an acyclic transaction graph G_t = {T,E}. T symbolizes a group of transactions saved in the Blockchain and E symbolizes the group of unidirectional edges among these transactions. The graph depicts the movement of Bitcoins among transactions as time passes. The group of inputs and outputs of coins can be thought as the weight in the edges of the graph. Specifically, every incoming edge e ∊ E in a transaction brings a timestamp and an amount of coins (Ci) which are the input for that transaction.

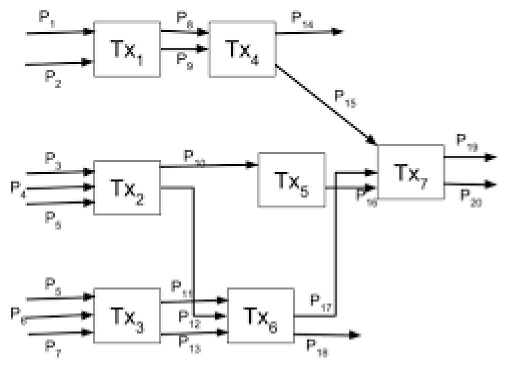


Figure 17: Transaction Graph [34]

2. Address graph: If we traverse the above graph, it not difficult to assume the connection among inputs and output public addresses. This connection will allow us to produce an address graph G_a = {P,E_1}. P symbolizes the group public addresses and E_1 symbolizes the edges that are linking these addresses

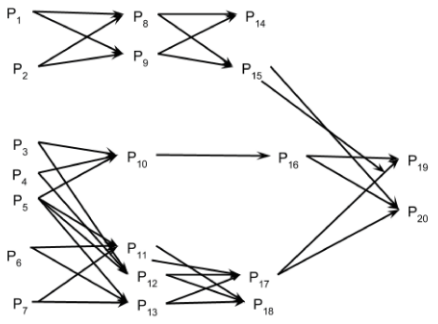


Figure 18: Address Graph [34]

3. User/entity graph: Having the address graph and some assumptions [14] which are generated from the Bitcoin protocol, the final phase is to produce an entity graph. This will be done by putting addresses in teams that seem to have the same owner. In the entity graph, Ge = {U,E_2} U symbolizes a disjoint subset of public keys (p) so that p ∈ P and E_2 symbolize the edges linking different U_1 in order to indicate a directed linkability among them

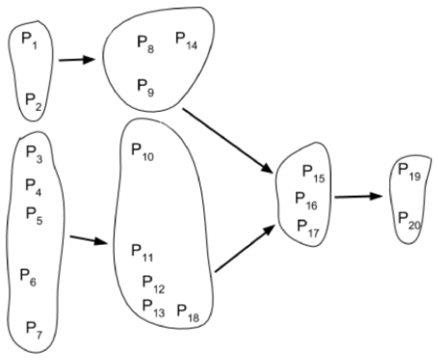


Figure 19: Entity Graph [34]

With the above methodology, the anonymity of the transactions in the network can be violated. So, we seek a solution that would enhance our anonymity and also be a trusted mechanism in a big scale. In order to make it more comprehensive, let's assume that we have Bob and Alice and they both control a virtual public address. Alice desires to buy a service or a digital product from Bob. So the goal for Alice is to transfer the indicated amount of coins to Bob's public address in an anonymous way. In other words, nobody should be able to link her to Bob, not even Bob himself. For this reason researchers have introduced mixing services or tumblers which may assure that it will transfer coins and randomly exchange them for other users' coins to obfuscate their ownership, although these come with no protection from theft by the service.

1.14 Problem Statement

Peers in cryptocurrency networks are represented with pseudonyms, which are their public keys. The transactions that are appended in the Blockchain depict the path of a cryptocurrency from a sender to a recipient. The path is very simple for a peer (malicious or not) to observe and connect the cryptocurrency to its initial possessor. This happens due to the public and transparent character of the Blockchain. Throughout the years, a great number of techniques have been created in order to deanonymize the peers' anonymity. Those kinds of attacks are particular danger because the public key of a user is connected to his real world identity. So, there is the need to improve and protect the peers' anonymity.

As defined above anonymity = pseudonymity + unlinkability. In the equation pseudonymity is not hard to achieve in contrast with the unlikability. Total unlikability is considered at least theoretically impossible to have, due to the fact the there is a probability 1 / n that a transaction can be linked to its peer. The variable n represents the anonymity and we consider that each of the clients in the set has the same probability to be the one triggering the transaction

2. TECHNOLOGIES

In this chapter we will analyze two anonymous networks that peers have attempted to utilize in order to enhance the anonymity of electronic transactions. The networks we will examine is the Onion Routing Network and the Invisible Internet Project

2.1 The Onion Routing Network

2.1.1 Defining the Onion Routing Network

The onion router [35] network is founded on the idea of onion routing. The network includes almost 6000 volunteer-operated routers, which are named as Onion Routers (ORs). Every OR generates a router descriptor that includes the personal data of the router. For example, these data consists of its IP address, public key, port as well as its bandwidth capabilities. Afterwards, the OR transfers the router descriptor to the directory authorities. The directory authorities create a network consensus report which is delivered with the descriptor to the directory servers. The Onion Proxies (OPs) which are the peers that run locally the Tor[36] software obtain the descriptors and consensus report from the servers. The elements that were downloaded from the proxies are used in order to create paths known as circuits, through the Tor network before they have the ability to communicate with their destination. Every circuit the most times is made from three ORs or hops. The ORs depending on their position in the circuit they are called entry OR, middle OR and exit OR. The onion routers maintain a connection with one another through TCP connections. The method of TLS is implemented in order to provide authenticity, data integrity as well as confidentiality. An example of Tor network is depicted in the below figure 20.

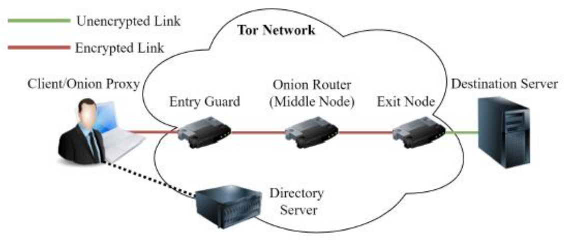


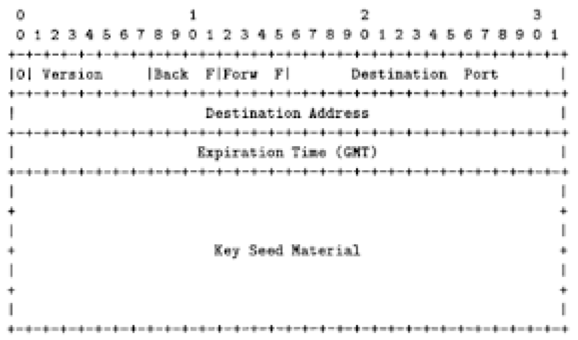
Figure 20: Tor Network[37]

2.1.2 Onions

The OP in order to create the anonymous connection from the entry to the exit node, an onion is generated. An onion is defined as a multilayered data structure that includes the path, beginning from the exit OR and going backwards at the entry OR. The structure of every layer in the onion is depicted in figure 21.

In order for the RSA public key cryptography to be executed properly, the first bit in every layer is required to be zero. After the zero field, the Version number of the onion routing network is placed. Following the Version number, a field named Back f is found. This field indicates the cryptographic function to be implemented to the onion which is travelling in the backward direction (when the onion is moving from the exit node to the entry or in other words is the opposite direction from which the onion moved initially).Moving forward, we see the Forw field, which shows the cryptographic function that must be implemented to the onion which is going in the forward direction (when the onion traverses the network from the entry to exit node).

Furthermore a Destination Address and a Destination Port field exist in the layer. Those fields clarify which will be the next OR that will be added in the network. In the case that the onion has reached the exit node both fields will be zero. The Expiration Time field is given in network order in seconds relative to 00:00:00 UTC January 1, 1970 (i.e., standard UNIX time (2) format) and specifies how long the onion router at this hop in the anonymous connection must track the onion against replays before it expires. Finally we observe the Key Seed Material field, which is 128-b long and is hashed three times with SHA in order to generate three cryptographic keys



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| Version     |Back  F|Forw  F|       Destination  Port       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Expiration Time (GMT)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Key Seed Material                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Onion Structure[35]

2.1.3 Directory Servers

The Tor network utilizes a small set of redundant, famous OR in order to be able to find any modifications concerning the network topology and the nodes states. Every directory server has a similar behavior with an HTTP server, so clients can obtain the latest router lists as well as the latest network state. Every onion router sends signed statements including personal information to directory servers periodically. When the directory servers receive the statement, they combine it with their own view of network liveness and they produce a signed description of the whole network state.

Also a directory server is able to verify, if the OR’s key is identified. This process is completed through the information that the directory server has obtained from the statements the onion routers have sent. If the confirmation gets rejected the servers won’t advertise unidentified ORs

It is very important for the directories servers to be synchronized and redundant, thus they can reach an agreement for a common directory. So, clients should only provide their trust to only this common directory because it will be signed by a threshold of the directory server

2.1.4 Cells

The ORs achieve communication among them through TLS connections with temporary keys. The communication inside those connections is depicted with fixed size cells. Every cell has a size of 512 bytes which includes a header and a payload. The header contains a circuit identifier (circID) which explains which specific circuit the cell is referring to and a command in order to make clear what the payload of the cell is going to do. It is important to underline that a great number of circuits can be created through a unique TLS connection. The cells can be distinct in two categories, which are the control cells and the relay cells. This distinction is made according to the command that is in the header.

2.1.4.1 Control Cell

This type of cells is always read from the OR that received them or from relay cells.In figure 22, a control cell is depicted. The control cell commands are the following:

- Padding: it is utilized for keepalive
- Create: establish a new circuit
- Destroy: ruin circuit

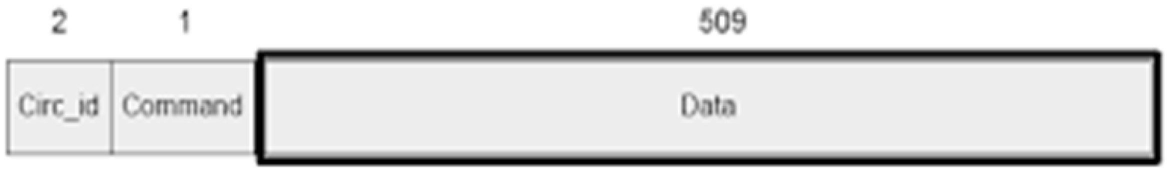


Figure 22: Control Cells

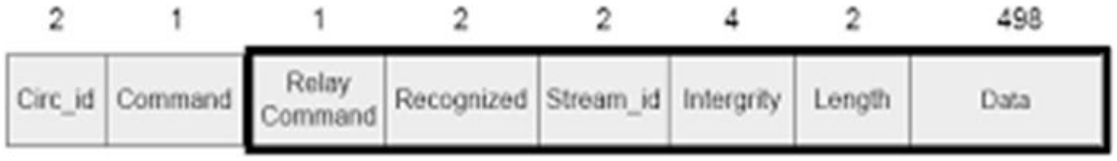
2.1.4.2 Relay Cells

The relay cells are transferring end-to-end stream data. They also have an extra header which is called the relay header. This header is placed at the front of the payload. The relay header consists of:

- streamID: the stream identifier
- end-to-end checksum: it is used in order to verify the integrity
- the size of the relay payload
- relay command

The relay header as well as the relay cell payload is simultaneously encrypted or decrypted as the cell traverses the circuit. In figure 23 a relay cell is depicted The commands of the relay cell are the following:

- relay begin: to open a stream
- relay end: to close a stream
- relay teardown: to close a broken stream
- relay connected: informs the onion router that a relay begin command has been performed correctly
- relay extend: to increase the circuit path by a hop and to acknowledge
- relay truncate: to teardown only part of the circuit and to acknowledge
- relay sendme: utilized for congestion control
- relay drop: utilized to apply long range dummies



2	1	1	2	2	4	2	498
Circ_id	Command	Relay Command	Recognized	Stream_id	Intergrity	Length	Data

Figure 23: Relay Cells[35]

2.1.5 Constructing a Circuit

The circuits are built from OP increasingly, who negotiate a symmetric key with every OR participating in the circuit. It must be underlined, that the negotiation is being implemented with one OR every time. The first step to build a new circuit is the OP (name her Alice) to transfer a create cell to the entry guard node in the selected path (name him Bob). Alice must pick a fresh circID C_{AB}, which is not currently utilized on the connection established from her to Bob. The payload of the cell that is sent from Alice includes the first half of the Diffie-Hellman handshake, which is encrypted from the onion key of Bob. Afterwards, Bob replies back by sending a created cell which includes a hash of the negotiated key. By the time the circuit has been created, Alice as well as Bob can send relay cells among them through the negotiated key.

In order for the circuit to be increased, Alice transfers a relay extend cell to Bob. The cell includes explicitly the address of the following OR and the first half of the Diffie-Hellman handshake which is encrypted from the onion key of the next OR. After, Bob makes a copy of the half-handshake inside a create cell and sends it to the indicated OR (lets name him Carol) in order to extend the circuit one hop .Bob must also select a new circuit ID which is not utilized currently among him and Carol. By the time Carol has sent a reply back to Bob with a create cell, he stores the payload inside a relay extend cell and sends to Alice. As a result from the above procedure Alice and Carol have a common key. Finally, the same process is followed in order to add a third OP in the circuit. Now that Alice has finished constructing the circuit, she has a common key with every OR. So she has the ability to send relay cells. The cells which are sent inside the network from Alice are encrypted with one layer of encryption for every OR that participates in the circuit using the correspondent key. As the cell passes through an OR one layer of encryption is decrypted.

2.1.6 Rendezvous Points

2.1.6.1 Defining Rendezvous Points

Rendezvous points are defined as building blocks for location-hidden services, which are also named as responder anonymity. This permits Bob to provide a TCP service, without exposing his IP address. Responder anonymity offers protection over DoS attacks because malicious peers must attack the Tor network due to the fact that they have no information about Bob’s IP.

We offer location-hidden services to Bob by permitting him to advertise a great number of ORs which consist of his introduction points. The introduction points are the contact points of Bob. Alice picks an OR to be her rendezvous points. After, Alice establishes a connection with Bob through one of his introduction points in order to reveal to him which OR is her rendezvous point

2.1.6.2 Rendezvous Points in Tor

The steps that follow are completed on behalf of Alice and Bob

- Bob produces a public key pair to recognize his service
- Bob picks several introduction points, and publishes them on the lookup service, and he signs the publishement with his key
- Bob creates a circuit to every introduction point that he has published, and the circuits are awaiting requests. Instead of being dependent of the external infrastructure, the Tor network executes the lookup service on its own. The present implementation offers a lookup system on the directory servers
- Alice is informed about Bob's service out of band. She obtains the information concerning Bob's service from the lookup service. In case Alice desires to get access to Bob's service anonymously, she must establish a connection with the lookup service through Tor.
- Alice selects an OR to be the rendezvous point. She constructs a circuit to the rendezvous point and provides it with an arbitrary selected "rendezvous cookie" to recognize Bob.
- Alice picks and establishes an anonymous connection to an introduction point that belongs to Bob and sends to it an encrypted message. This message includes details for her rendezvous point, "rendezvous cookie" and the beginning of a DH handshake. The introduction point forwards the message to Bo
- If Bob decides to communicate with Alice, he constructs a circuit to Alice rendezvous point. He will send a message including the rendezvous cookie, the second half of the DH handshake, and a hash of the session key.
- The rendezvous point links Alice's circuit to Bob's.
- Alice transfers a relay begin cell inside the circuit. It goes to Bob's OR, which links to Bob's webserver
- An anonymous stream has been created

2.1.6.3 Goals of Rendezvous Points

- Access-control: Bob requires a technique to filter arriving requests, so malicious peers are unable to flood Bob by creating a lot of connections to him.
- Robustness: Bob must be capable to retain a long-term pseudonymous identity even if a router failure happens. The service that Bob is providing must not be tied to a specific OR but instead he must be able to resettle it to other Ors that belong in the network.
- Smear-resistance: A social attacker must not have the ability to "frame" a rendezvous router by providing an illicit or notorious location-hidden service and tricking observers believe that the router generated that service.
- Application transparency: Despite the fact that peers are obligated to use specific software in order to have access to location hidden-services, we do not demand from them to change their applications

2.1.7 Router Selection Algorithm

2.1.7.1 Limitations

The initial Tor concept supported that ORs are picked in a random way for all circuits. So this would give the same probabilities to all routers to be selected. Also in this way the level of complexity for an attacker to succeed in a deanonymize attack in order to expose the ORs participating in the circuit, increases. But, due to the issue of the heterogeneity in resources the protocol was modified in order to satisfy the below limitations.

- All onion routers must appear only one time in a path and simultaneously onion routers that belong in the same family cannot be in the same circuit. Coadministered onion routers can be marked as belonging to the same family by operators to avoid hindering users' privacy.
- Onion routers are being evaluated from directory authorities according to their performance, stability and role in the network. This evaluation happens with the distribution of flags to routers. For example if an onion router has an active role inside the network, and its bandwidth is in the best 7/8 of known active onion routers, it gets marked with a green flag
- In order for Tor to develop some defense mechanisms against a variety of attacks such as finding hidden services, the way Tor picks the onion routers changed, in order for the entry node to be selected from a set of nodes named entry guards. This type is a node whose weighted fractional uptime is at least the median for active routers, and its bandwidth is at least the median.
- Picking an onion router in order to be placed in the path is relative to the bandwidth it provides. This happens so the most capable and efficient routers are selected with a higher frequency

2.1.7.2 Entry Router Selection Algorithm

The original algorithm utilized to pick entry routers was changed in May 206 with the release of Tor version 0.1.1.20. Entry guards were imported in order to enhance the defense, mechanisms in order to face disruption attacks .Using this method it is decreasing the possibility of an attacker purposely breaking circuit until they control an aimed circuit [37]. This selection algorithm operates by automatically choosing a group of Tor OR that, are indicated as fast and stable. Fast routers are the routers that advertise bandwidth above the median of all bandwidths publications. A stable OR is characterized as the one that advertises an uptime that is higher than the median uptime of all others.

The OP will only pick fresh entry guards when one is unreachable. In the time speaking, the default number of entry guards picked is three, and old entry guards that have not succeeded are saved and retried. There is also an additional choice, which is to utilize only the entry guards that are hard-coded into the configuration file. This algorithm was applied to provide protection in the initial hop of a circuit by utilizing a limited pool of nodes.

2.1.7.3 Non-Entry Router Selection Algorithm

The second algorithm to choose non-entry nodes is aiming to optimize OR selection for bandwidth and uptime, while not always picking the very best nodes every time. This is meant to assure that every node in the network is utilized to some degree, but nodes with more bandwidth and greater stability are utilized more frequently. Tor owns a group of TCP ports that are labeled as “long-lived.” If the traffic transiting a route utilizes one of these long-lived ports, Tor will optimize the route for stability by shortening the list of available routers to only those that are labeled as stable. This makes Tor’s routing algorithm to prefer routers that are labeled as stable nodes. In order for somebody to comprehend in more depth the algorithm, see the Tor Path Specification [38]

2.1.8 Tor Weaknesses

2.1.8.1 Scalability

Scalability is the ability of the network to increase as the peers participating in the network increase. The scalability problems exist due to two primary reasons. The first one is that Tor has a centralized nature and the second one is the high client-to-router ratio. The centralized construction is utilized in order to assist OP and OR to bootstrap as well as help them to maintain a renewed list of OR which serve the network. This is essential because Tor can face attacks concerning portioning attempts which continuously leverage data in order to violate the anonymity of peers. Let’s assume that OP is utilizing an old group of OR, whose size is not bigger than the number of current routers this will result in providing to attackers a higher probability to guess the routers that are part of the circuit. So, OR as well as OP download the router descriptor and the consensus report. In this way they have access to all onion routers any time in order to maintain a stable view of the Tor network. The centralization may provide strong defense mechanisms but is a main reason for scalability issues

2.1.8.2 Security

In order for the anonymity in Tor to be violated the attacker must be able to be in charge of the entry and the exit node of the circuit. Through the last years several attacks have been suggested in order for attackers to increase the capability of compromising nodes. Some well known examples are the Selective Denial of Service (SDoS) attack and the side-information attacks.

2.1.8.3 Circuit Construction

In the onion routing network, to build a circuit, the OP produces an onion where every layer encapsulates the key for the corresponding onion router as well as information for the proceeding router on the route. The primary issue in this technique is that it does not offer security. In other words, if an onion router is compromised, than it is possible to expose information concerning past user information. This attack can be implemented, if two conditions are satisfied by the attacker. The first one is, that he must keeps track of all communication in the network. The second is to achieve to get a private key of an OR, in order to utilize it to decrypt the session key data, so subsequently he will be able to decrypt the remaining communication data. But in order for Tor to be able to face this attack the circuit is constructed increasingly and interactively as described above. It must be underlined, that this method is very expensive because it demands network communications as well as encryption and decryption methods.

2.1.9 Attacks against Tor

2.1.9.1 Defining Attacks

The attacks described below, demand from the attacker to be able to access the entry as well as the exit OR of a circuit. This can be done by compromising Tor nodes that exist in the circuit or by importing in the circuit nodes that are controlled by him. When importing new nodes, specific steps can be implemented to augment the probability to be chosen either as an entry node or an exit node. The ORs can indicate that they are available only if they take the role of the exit node in the circuit. This will be done by configuring exit policies to permit specific protocols. Also, a node can publish bandwidths as well as uptime with bigger value in order to be picked as an entry node. By the time the attacker has access to both ORs, the traffic information needed to launch an attack are transferred to a central entity for processing.

2.1.9.2 Low Resource Routing Attacks Against Tor

One of the most known attacks that were created in order to violate the anonymity in Tor circuits was in 2007 by Bauer [39]. This attack method consists of 2 phases. In the first phase, the attacker takes for granted that he is in charge of a big number of OR.As mentioned above, this can be fulfilled by either importing new malicious nodes or by compromising the nodes that are inside the network. In the first versions of Tor, ORs were able to publish a false bandwidth and uptime to the directory server. During the process that the circuit was generated no entity confirmed the values that the ORs had advertised. So the resources needed for this method could be decreased by publishing fake bandwidths for low bandwidth connections and at the same time increasing the probabilities for the node to take the role of an entry or exit router in the circuit. Furthermore, the attacker's nodes were able to have no limitations on their exit policies in order to make it more possible to be picked as exit nodes. If only one malicious node is participating in the circuit than it can block the movement of the traffic in order to compel the circuit to be reconstructed with a different group of nodes. This process will consistently be executed from the malicious node until two malicious nodes are part of the circuit, which it is obligatory that they are the entry and the exit node.

The second phase of the attack demands traffic correlation. Every malicious OR inside the network must register information for all cells received. This information contain the position of the cell, the timestamp, the preceding link's IP address as well as port and the following hop's IP address as well as port. Finally, the central entity which obtain the above information from every malicious router can perform a correlating algorithm to link the sender with the recipient

2.1.9.3 Protocol Level Attacks

Fu imported a new type of attack, the protocol level attacks [40]. These attacks can be implemented by manipulating only one cell inside the circuit. The attacker must be able to control the entry and the exit onion router utilized by a specific circuit. The malicious entry router records information which consist of the source IP address and the port utilized for a particular circuit, the circuit ID, and the time of the cell being manipulated.

Then the attacker is ready to execute the attack, which can be implemented with the below ways:

- duplicating an aimed cell along the specific circuit and then forwarding the duplicated cell
- changing some bits of 509-bytes data of an aimed cell and forwarding the modified cell to the succeeding hop
- importing an artificial cell inside the target circuit at an suitable time
- erasing a target cell without sending it to the succeeding hop

The duplicated cell, changed cell, artificially imported cell, or the cell following the erased cell crosses the circuit and arrives at the exit router. The attacker at the exit onion router due to the fact that it is controlled by him, he can distinguish cell recognition errors caused by those manipulated cells. After, the malicious peer stores the time of the cell recognition error, the receiver's IP address and port linked with the circuit, and the corresponding circuit ID. The above process provides the attackers the ability to verify that the aimed cell gets in from the entry node as well as that the same cell goes out from the exit node. In conclusion, by the time the entry router has knowledge of the OP IP address of the TCP stream and the exit onion router is informed of the destination IP address of the TCP stream, the communication among the entities will be verified

2.2 The Invisible Internet Project

2.2.1 Defining the Invisible Internet Project

The target of the Invisible Internet Project (I2P) [41] is to offer to peers the right of anonymous communication in an overlay network over the Internet. In other words, through this network no useful metadata can be taken from a message. So the parties that communicate will keep their identity anonymous. This target must be accomplished by generating a network which consists of routers. The routers are nodes among which packets can be transferred. In order to offer anonymity to all peers, a message is traversing through a lot of peers, making it complicated for an attacker, who observes data traffic, to link the sender with the recipient. Due to the fact that the message includes important information, it is encrypted before it is sent. Finally, it must be underline that the idea of this network is an extent of the Tor network.

2.2.2 Tunnels and Tunnel Creation

An I2P Tunnel is defined as a group of routers in predefined order utilized to forward a message. It normally includes a gateway (the initial router of the tunnel), a participant (the router in the middle) and an endpoint (last router in line). The messages are sent through outbound channels and are correspondingly received through inbound channels. Every client generates and maintains more than one tunnel at the same time.

The generation of an outbound tunnel is depicted in figure 24. The router that aims to create a tunnel obtains a list of routerInfos with potential peers from the NetDB (it will be described in later section), picks the wanted number of routers (often 2 but they can be more) based on their capabilities and places them in preferred order. After, it executes computations in order to create messages for every user in the tunnel where itself behaves as a gateway .Then the messages are transferred to the succeeding node which is informed that a tunnel must be generated in order to forward the message to the endpoint. When this is done, it waits for the acknowledgement from the endpoint and transfers it to the tunnel generator. If everything goes well, the tunnel can then be utilized to send messages to other peers

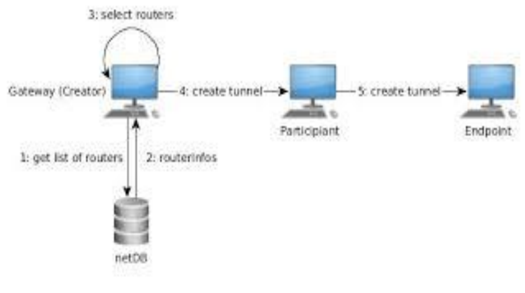


Figure 24: Creation of an outbound channel[41]

The generation of an inbound channel follows the same procedure as the creation of an outbound channel. The only difference is that the creator behaves as the endpoint and the last router in the network becomes the gateway. A Tunnel-ID is utilized so, the gateway can connect the received messages to the correspondent receiver. Finally, the creator broadcasts the address of the gateway as well as the Tunnel-ID in the NetDB In figure 25 the creation of an inbound channel is depicted

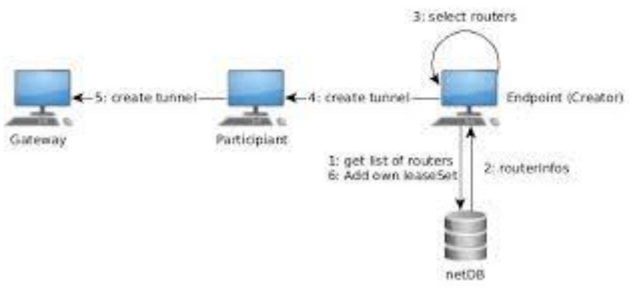


Figure 25: Creation of an inbound channel[41]

2.2.3 Message Structure

Data inside the I2P network is transferred in the form of I2P Network Protocol messages. They primarily include delivery commands and payload, which may contain the full message or a part of it, because of size constraints. In order for the network to defend against Timing attacks, a great number of messages for a particular router are mixed into a Tunnel Message. This message includes the ID of that router, the IV used for encrypting the payload. The payload with its turn consists of a checksum, padding (if it is necessary) and the collected messages for the router in the form of “Delivery Instructions” (where to next send this message) and the I2NPs itself (mentioned as ”Clove”).In Figure 26 is depicted the Tunnel Message structure. The whole specification of the Tunnel Message is placed in the I2P Documentation [10]

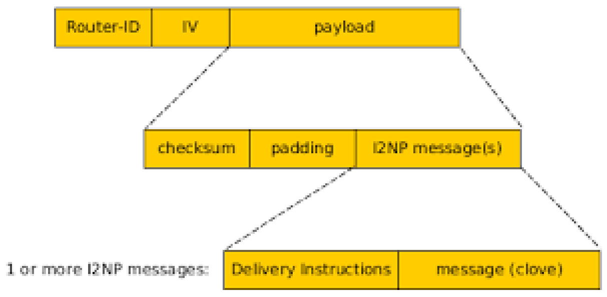


Figure 26: Message Structure[41]

2.2.4 NetDB

The I2P NetDB is defined as a database that includes information about all peers in the I2P network. It controls two sets of information required to function the network:

- routerInfo: Includes the information concerning a router, such as its Identity (the ElGamal key, the signing key and certificate) or how to come in touch with it (it’s public IP address and port). This information is delivered from the router to the NetDB as soon as it links to the network and is recognized by the SHA256 hash of the Identity.
- leaseSet: A leaseSet includes the information demanded to contact a particular service, containing a list of potential tunnel entry points (router IDs of the gateways and tunnel IDs of its inbound tunnels), the Identity (ElGamal key, signing key, certificate) and possibly additional encryption or signing keys. The

leaseSet is also recognized by its destination, the value of the SHA256 hash of the Identity.

2.2.5 I2P Network

The I2P network consists of I2P routers that transfer encrypted garlic cloves and I2P peers that broadcast and obtain such cloves to one another. I2P routers and end-user client nodes have specific identifiers through cryptographic identities, which allow them to transfer and obtain messages as well as generate encrypted tunnels. Every I2P node inside the network creates inbound and outbound tunnels to other I2P routers. The tunnels can be augmented in order to generate different paths by just linking to different I2P nodes. When a message is required to be transferred from a sender to a receiver, the message traverses through the sender's outbound tunnel where in the end of it, exists another I2P node. In this point of the message transfer, two scenarios can take place. The first one is that the node will forward the received message through subsequently hops to its destination. The second scenario is that it will send the message directly to the recipient. In both cases in the end the message will be force to travel through the recipient's inbound channel and when it reaches at the recipient's node it will get decrypted. The senders have no knowledge concerning the path that the message will follow in order to reach the recipient except of course the first I2P node. In figure 27 it is depicted an I2P network.

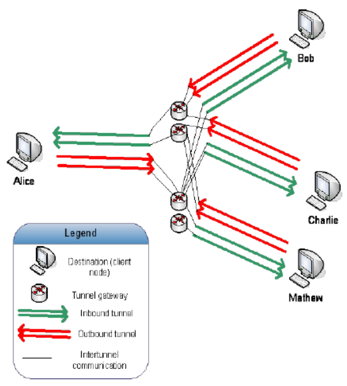


Figure 27: I2P network[41]

3. MIXING PROTOCOLS

In this last chapter of the thesis we will analyze in depth the main method that is being utilized in order to enhance the anonymity in the electronic transactions which is known as mixing services. It will also be provided an evaluation of every mixing protocol of all categories. Finally, it will be explained how every previous mixing model assisted in the development of a following mixing protocol

3.1 State of the Art

The most famous technique in order to enhance the privacy that is provided to the peers is through coin mixing or tumbling. Mixing is the service which will assure that it will take the coins of a peer and randomly swap them for other peer's coins in order to obfuscate the identity of their holder. This method offers k-anonymity. The concept is that k peers, all deposit one coin and then with the assist of mixing protocols the coin is mixed and transferred to its recipient. This method provides peers strong privacy and anonymity due to the fact that the coins are impossible to be connected to them. Many mixing protocols have been proposed which are centralized, decentralized or implemented by a smart contract

Centralized mixing is when the mixing is implemented by a third party (trusted or untrusted) called mixer or tumbler. Decentralized mixing is when there is no use of a third party to implement the mixing but in the contrary, the mixing is done by the participants. An important drawback of centralized mixing protocols is that its functionality and safety is linked with the trusted party. In the contrary, decentralized mixing protocols provide strong theft prevention and anonymity but they demand large computational power.

Smart contracts are an agreement between disbelieving parties who desire to participate in a mixing transaction. The contract is executed from the Blockchain consensus protocols. The Ethereum network is the most known framework in which smart contracts are used.

3.2 Centralized Mixing Protocols

3.2.1 Mixcoin

In the Mixcoin [42] protocol a client Alice, possesses a number of coins at a public input address. This public address is connected to her real world identity. Alice desires to send a part of her to coins to a public output address which will be complicated for any peer to connect it to her input address. This process will be completed only if Alice pays the mixing fees that correspond to her. So, Alice deposits her coins to a mixer. The mixer will keep her coins in an escrow address for an agreed time. Finally the mixer will transfer the coins to the requested output address, before the agreed time has expired.

3.2.1.1 Core Protocol

In the beginning the client must communicate with a mixer and propose to him the initials values of the mixing parameters. The mixing parameters that must be specified are the following:

- u: the amount of coins that will be mixed
- t_1: until this time the client must have transferred the agreed amount to the mixer
- t_2: up to this time the mixer must have transferred the agreed amount to the output address
- k_{out}: the client's output address
- r: the mixing fee that the mixer will be paid by the client
- n: a nonce that is used in the Beacon function
- ω:the number of blocks that are needed to confirm that the client has paid

The mixer will either decline or accept the terms. In the first case the protocol will be aborted. The client will search for another mixer, which will accept his terms. In the second case the mixer will produce a new escrow address, in which he will receive the client's coins. Afterwards, he will return a warranty signed by its private key, which will contain the client's mixing parameters and the escrow address.

Until this phase, the client has not yet made any promise to the mixer that he will pay. So, in case he changes his mind or misses the agreed time (t_1), by which he must transfer the coins, the protocol will be aborted. The mixer will just erase any record of this transaction. If the client makes the payment on time (t_1), then the mixer is obligated and has no other choice but to transfer the same amount of coins, by the agreed time (t_2) to the client's output address (k_{out}). There is a possibility that the amount will be kept by the mixer as a mixing fee. This is determined by the Beacon function. Lets' assume that the client transfers the money to the mixer on time.

If the mixer behaves in an honest way, then the transaction will be published on the Blockchain and both the client and the mixer will erase the records they have saved. In the contrary, if the mixer misbehaves either by not sending the money to the indicated output address (k_{out}) or not sending the money by the agreed deadline (t_2), then the client will publish the warranty. Due to the fact that the warranty is signed with the mixers private key and the Blockchain is public, every peer will be able to verify the misbehavior of the mixer. This will have as a result to decrease the mixers reputation and it will not be chosen again by the peers in any transaction.

It must be underlined that the mixer controls multiple escrow addresses. This means that the escrow address in which he receives the coins from the client can be different from the escrow address from which he transfers coins to the indicated output address

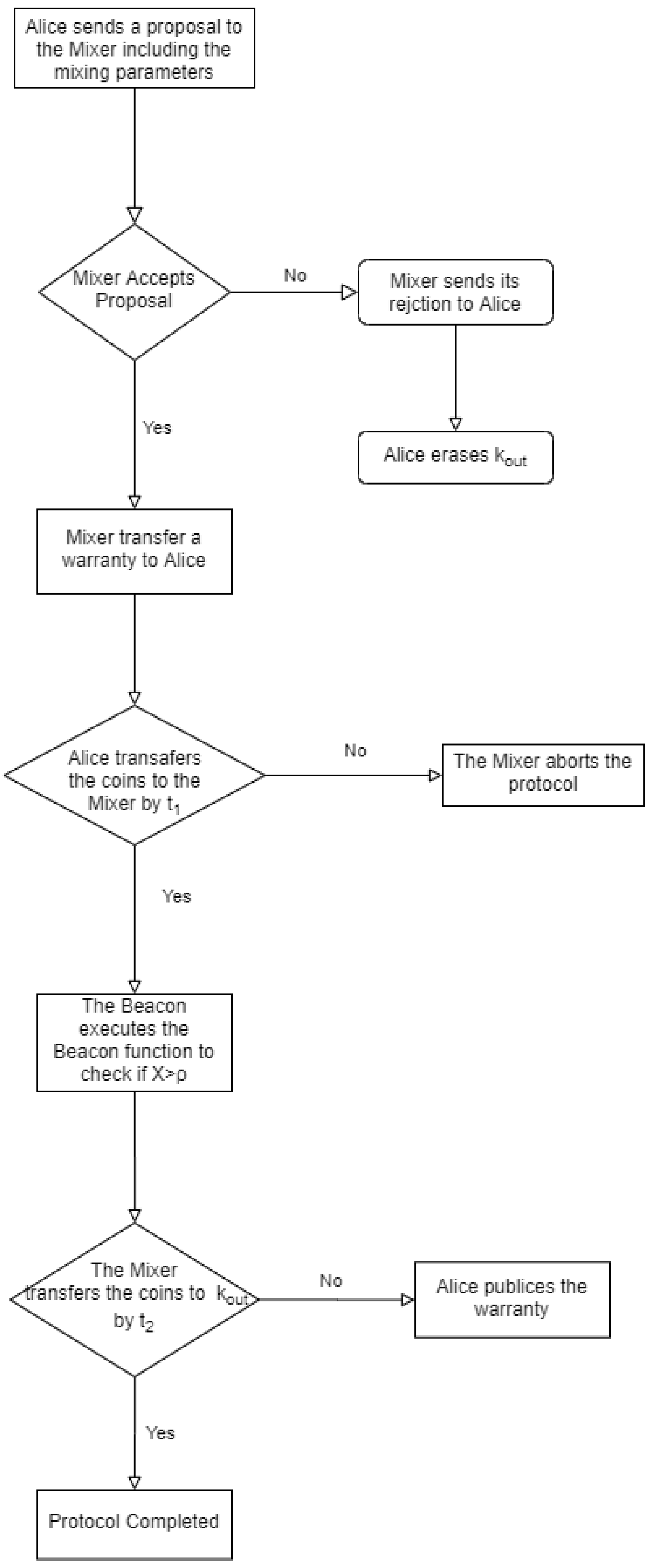

Alice sends a proposal to the Mixer including the mixing parameters
Mixer Accepts Proposal
No
Mixer sends its rejction to Alice
Alice erases k_{out}
Yes
Mixer transfer a warranty to Alice
Alice transafers the coins to the Mixer by t_1
No
The Mixer aborts the protocol
Yes
The Beacon executes the Beacon function to check if X>ρ
The Mixer transfers the coins to k_{out} by t_2
No
Alice publices the warranty
Yes
Protocol Completed

Figure 28: Mixcoin Protocol

3.2.1.2 Freshness of Address

The above protocol was described for the case that a client needs only one mixing round in order to transfer the total amount in the output address of its recipient. There is the case where the client needs more than one mixing round (if the value of k-anonymity is smaller than u). In every mixing round, it is essential that the escrow address and the output addresses that are used by the mixer and the recipient correspondingly are changed. This requirement is resulting from the fact that none of the input address and the escrow address (the one from which the mixer will complete the payment) are included in the warranty, so the warranty will be fulfilled from the moment that the money are transferred to the escrow address (the one in which the client will transfer the money) and after to the output address. The warranty takes no notice from which address the money came. So the client and the mixer must choose addresses with no other possible source of income. This will assure that the parties indeed paid and not a different entity made the payment. Also, less information will be exposed if the client publishes the warranty (because the output address participates in a transaction that has failed).

3.2.1.3 Mixing Fee

An easy way to define a mixing fee rate r is to specify the fee amount and force the mixer give back (1-r)*v to the output address instead of the total amount u. But this would cause a lot of problems in the case of sequential mixing due to the fact the smaller output value (1-r)*v is not in position to be the input to a subsequent round of mixing with the same u.

So Mixcoin protocol proposes the method of randomized mixing fees. In this method the mixer will ether maintain the total amount of coins that the sender will sent or he will transfer the whole amount in to the indicated output address. This is determined by the execution of the Beacon function. So, this generates an expected mixing fee rate of r and leaves the output address with either any coins or a complete u which than it can be re-mixed.

3.2.1.4 Generalization

Before Mixcoin protocol was designed a lot of mixing websites existed such as Bitcoin fog and BitLaundry. Every one of them provided the operation to mix transactions anonymously in exchange with some service fees. These websites behaved as online mixers and traded transactions between different peers, so they can break the link among incoming and outgoing transactions. In the above services there were two main issues:

1. The website that offered the mixing service could be the malicious entity. So, it would not complete the transfer and instead it would keep the coins. Simultaneously users don't have any way of preventing the steal or proving that the service provider is malicious
2. The mixing website is placed between the sender and the recipient, so it always maintains a record for a certain time. This has as a result that the mixing website is able to route the transactions inside the system. At the same time, users have no guarantee that their personal information will not be disclosed

So Mixcoin was introduced to the mixing community in order to face the first problem, by providing evidence that the middle entity misbehaved and did not operate according to the protocol. The Mixcoin achieved this through a signature method which provided accountability. So the user has the ability to publish in the network the evidence and decrease the Mixers reputation.

3.2.1.5 Advantages

- The Mixcoin protocol provides to the mixing peers the service of accountability. More specifically, if the mixer tries to misbehave either by not sending the money to the indicated output address or by not sending until the agreed deadline, then the client will publish the warranty. Due to the fact that the warranty is signed with the mixers' private key and the Blockchain is public, every peer will be able to verify the misbehavior of the mixer.
- The anonymity is protected and maintained against senders, recipients and outsiders. In more detail neither the senders nor the recipients can link transactions in specific peers. This happens due to the k anonymity technique and the mixer which break the links.
- The Mixcoin protocol provides mix indistinguishability. The majority of senders in order to transfer the whole amount of coins will participate in more than one mixing rounds as well as mixers. Thus, an adversary will not be able to connect the transactions implemented to specific mixers
- This protocol has the ability to resist to denial of service attacks (DoS). In the majority of mixing protocols an adversary can DoS the whole mixing procedure by being part of the mixing until a certain point. Then he denies transferring the coins and the whole process collapses. Instead, in this protocol each peer communicates only with the mixer so, by denying to form into line with the protocol doesn't affects the other peers. Another case is for an adversary to implement such an attack by attempting to block transactions from being stored on the Blockchain timely. Those attacks are very costly and hard because the adversary must possess a large amount of the block mining pool.
- The Mixcoin protocol provides scalability. There are no performance issues by adding more peers to a mixing operation, due to the fact that peers communicate exclusively with the centralized mix and not with one another. Furthermore, if all peers use the same mixer there is possibility that a bottleneck may occur. Then there is the choice to load balance the operation of the mix on to deferent mixers, which all are functioning with the same cryptographic keys.

3.2.1.6 Disadvantages

- Mixcoin protocol doesn't provide a mechanism that assures the peers that the mixer is impossible to steal the coins. Even if the mixer's reputation is decreased, they would have already lost their money and won't be able to regain them
- The anonymity of the transactions can be violated by the third party due to the fact that the client sends him the output address that the coins must end up. So, he knows the pairs inputs-outputs
- The degree of anonymity depends on the number of peers participating in the mixing procedure. More peers mean that there is a smaller probability of a peer to guess the connection among inputs and outputs

3.2.2 Blindcoin

At a high level, the Blindcoin [43] protocol functions as follows. In the beginning, the mixer publishes the mix parameters, and then the client makes a proposal to the mixer, which will either be accepted or decline. After, the mixer gives to the client a partial warranty. By the time the client receives it, he transfers the coins to the mixer's escrow address. Afterwards, the mixer completes the warranty by appending it to the public ledger. In the next step the client unblinds the output address. Finally, the mixer sends the coins to the indicated output address.

3.2.2.1 Core Protocol

In the beginning a client must communicate with a mixer and propose to him the initial values of the mixing parameters. The mixing parameters that must be specified are the following:

- u: the amount of coins that will be mixed
- t_1: until this time the client must have transferred the agreed amount to the mixer
- t_2: until this time the mixer must have published to the Blockchain the client's token
- t_3:until this time the client must have unblinded the output address from the blinded token
- t_4: until then the mixer must have completed the payment to the indicated output address
- r: the mixing fee that the mixer will be paid by the client
- ω:the number of blocks that are needed to confirm that the client has paid

Along with the mixing parameters, the client also sends a blinded token which includes the output address and a randomly selected nonce which will be utilized in the procedure of fee collection. The key to keep the values private is to encrypt the token with an encryption function. This encryption function and the decryption function is only known by the client

The mixer will either decline or accept the terms. In the first case the protocol will be aborted. The client will erase the blinded token and he will search for another mixer which will accept its terms. In the second case the mixer will send back to the client a partial warranty signed by his private key. The partial warranty includes the mixing parameters, the blinded token and an address which belongs to the mixer's escrow address where the client must transfer the coins which will be mixed.

Up to this stage, the client has not made any promise to the mixer that he will pay so, in case he changes his mind or misses the agreed time (t_1) by which he must transfer the coins, the protocol will be aborted. The mixer will just erase any record of this transaction. If the client makes the payment on time (t_1) then the mixer is obligated to transfer the same amount of coins by the agreed time (t_4) to the client's output address.

By the time the client has transferred the coins to the mixer's escrow address, the mixer must complete the partial warranty that he has sent previously. This will be fulfilled by signing the client's blinded token and publishing it in the Blockchain timely (t_2). This mixer's action will acknowledge that the client has done the payment according to the terms that where agreed. Due to the fact that the Blockchain can be publically viewed it can be verified by any peer.

There is always a possibility that the mixer doesn't achieve to publish the signed token. That would result in failing to complete the partial warranty. In this case the client's action is going to publically incriminate the mixer and decrease his liability. This will be implemented by publishing some information that would function as evidence for the other peers. So the information that would be published in order to incriminate the mixer includes the following:

- the partial warranty that has signed with the mixer’s public key
- the transaction that indicates that the client transferred the money at the mixer. This is possible because the transaction was appended to the Blockchain
- the absence of the signed blinded token in the Blockchain

Using the above, any third party can verify the misbehaving of the mixer and this would result that no peer would choose this particular mixer in future transactions.

Let’s assume that the mixer follows the mixing process and by the indicated time he has published the blinded token at the Blockchain. Then the client must unblind the output address in order for the output address to be revealed, but this must take place anonymously. The client, in order to take back the signed unblinded token, she will apply the decryption function. It is of great importance that the client connects to the network with a different public address from the one that created the blinded token. The peers might connect through a safe anonymity network, for example Tor, and achieve to maintain their two different addresses unlinkable. In the reality, this is not easy to achieve against strong adversaries. After the decryption the client posts the token to the Blockchain.

After the unblinding and the posting are done on time (t_3), the mixer can be certain that the output address is correct because the token is a proof-of-work, due to the fact that the mixer has signed it with his private key. Once more, there is the possibility that the client doesn’t unblind and post the token. In such a case, the mixer has two choices. The first is to transfer the coins back to the client and the second is to keep them. In the second case the client is not in the position to incriminate the mixer due to the fact that the protocol breach was done from his side.

Next, the mixer must compute the Beacon function in order to decide if the transferred amount of money will be sent to the output address or will be kept as a mixing fee. If the computed value of the function is greater than the coins transferred it doesn’t keep them as a mixing fee. The mixer will transfer the coins to the output address if it acts in an honest way. But there is always the possibility that the mixer misbehaves by keeping the coins. In this case the client will try, as before, to incriminate him and decrease his liability. So he will publish at the Blockchain the following information:

- the partial warranty that has been signed with the mixers public key
- the transaction that indicates that the client transferred the money at the mixer. This is possible because the transaction was appended to the Blockchain
- the encryption function
- the decryption function
- the signed token
- the absence of the final transaction between the mixer and the recipient

Using the above, any third party can verify the misbehavior of the mixer and as result no peer would choose this particular mixer in future transactions

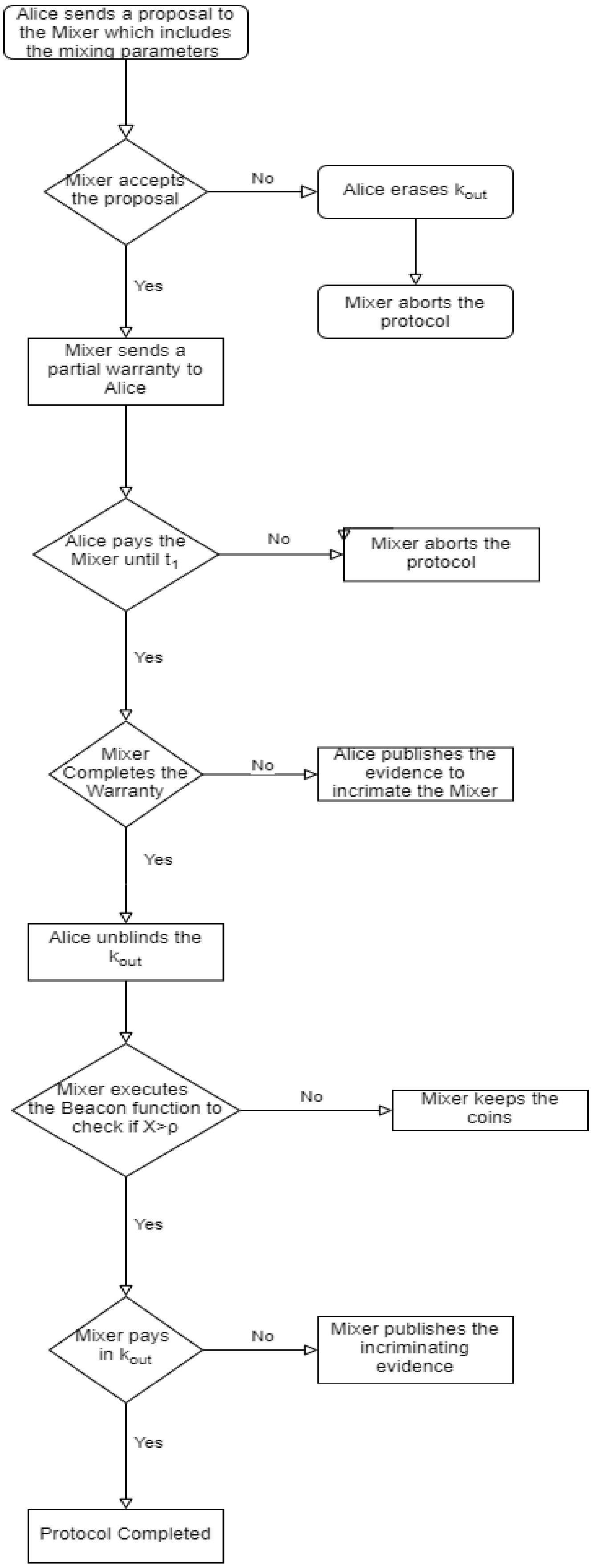

Alice sends a proposal to the Mixer which includes the mixing parameters
Mixer accepts the proposal
No
Alice erases k_{out}
Mixer aborts the protocol
Yes
Mixer sends a partial warranty to Alice
Alice pays the Mixer until t_1
No
Mixer aborts the protocol
Yes
Mixer Completes the Warranty
No
Alice publishes the evidence to incrimate the Mixer
Yes
Alice unblinds the k_{out}
Mixer executes the Beacon function to check if X>ρ
No
Mixer keeps the coins
Yes
Mixer pays in k_{out}
No
Mixer publishes the incriminating evidence
Yes
Protocol Completed

Figure 29: Blindcoin Protocol

3.2.2.2 Generalization

Blindcoin was founded on top of the Mixcoin protocol in order to overcome the vulnerability of the Mixcoin concerning the anonymity of peers. Simultaneously it provided a solution for the website mixers concerning problem 2. In more detail, as explained in the Mixcoin section the anonymity in the protocol can be violated by the Mixer because he has knowledge of the link among inputs and outputs. So Blindcoin imported the blind signature technique and he converted the protocol to be fully anonymous. As mentioned in Chapter 1 a blind signature is defined as a digital signature where the context of the message is blinded before it is send to the recipient

3.2.2.3 Advantages

- The Blindcoin protocol provides the mixing peers the service of accountability in two cases, where the mixer can violate the protocol. The first scenario is that the mixer never publishes the signed token and fails to complete the partial warranty. Then the mixer will publish the partial warranty and the evidence to incriminate him. The second scenario is that the mixer keeps the money or doesn't implement the transaction by the indicated time, than the sender publishes once again the evidence (the complete warranty) in order to decrease his liability.
- In every possible scenario, the linkability, the anonymity of the peers is protected due to two variables. The first one is the use k-anonymity by which it is not possible to distinguish the possessors of the transactions by observing the flow of coins in the Blockchain. The second one is the blind signature technique, so not even the mixer knows the output address of a sender, because the token that includes it is blinded. When the unblind takes place, the sender connects to the network with a different identity which cannot be linked to his initial, which makes it impossible once more for the mixer to break the anonymity.
- This protocol has the ability to resist to DoS. In the most mixing protocols an attacker can DoS the whole mixing protocol by participating in the mixing process up to a specific point, and then he refuses to transfer the coins and the whole process is blocked. In this protocol, each peer interacts only with the mixer so denying behaving as the protocol indicates doesn't affects the other peers. Another case is an adversary to implement such an attack by attempting to block transactions from being stored on the Blockchain on time. Those attacks are very costly and hard because the adversary must possess a large amount of the block mining pool
- The Blindcoin provides scalability due to the same reason that was mentiond above in the Mixcoin protocol.

3.2.2.4 Disadvantages

- Blindcoin protocol doesn't provide theft preventions to peers. In more detail during the operation of the protocol the mixer is able to steal coins from users. The only consequence that it will face is the decrease of his reputation
- Blindcoin protocol doesn't provide mix indistinguishability because when the unblinding and publishing of the signed token takes place, every peer by observing the Blockchain can connect the token to the mixer. This happens due to the fact that the token is signed by the mixers private key

- In the Blindcoin protocol the anonymity depends on the number of a peer that is participating in every mixing round because the probability of guessing the link among inputs and outputs is smaller

3.2.3 TumbleBit

TumbleBit [44] is a mixing protocol that targets in replacing on-Blockchain transactions with off-Blockchain transactions. This is achieved through puzzle solving protocols in which a sender pays the recipient by offering the answer to the puzzle. The puzzle is produced through the communication among the sender and the recipient. The solution to the puzzle is found through a communication among the sender and the Tumbler. Every single time that a solution to a puzzle is found, 1 coin is transferred from the sender to the Tumbler and subsequently to Bob. The sender establishes a payment channel with the Tumbler through escrowing Q coins on the Blockchain. In the same way every recipient establishes a payment channel with the Tumbler. When the channel is opened the Tumbler escrows Q Bitcoins on the Blockchain and after the Tumbler and Bob both participate in a puzzle promise protocol that produces up to Q puzzles for the recipient. As the off-Blockchain Payment stage takes place, every client fulfills up to Q off-Blockchain payments to any group of recipients. In order for a payment to be completed, the sender communicates with the Tumbler to learn the answer to the puzzle that the recipient offered. Last the Cash-Out phase takes place where all channels that were created are closed. If any coins have remained in the escrow addresses they are redeemed by the owner.

3.2.3.1 RSA-Puzzle-Solver-Protocol

The RSA-PUZZLE-SOLVER-PROTOCOL is used in the interaction between the client and the Tumbler in order to solve a puzzle z. Let's assume in our case that a client seeks the solution of a puzzle z and for exchange will give 1 Bitcoin to the Tumbler. At first the client receives the blinded puzzle from the recipient. Then he transfers the puzzle to the Tumbler in order to solve it. The Tumbler computes the blinded solution e of the puzzle z. He encrypts the solution e under a key k creating a cipher text c. After, the Tumbler hashes the key with a value h. Finally he sends the pair (c,h) to the client. The client in order to decrypt the solution, he needs the value of h. So he publishes on the Blockchain a T_{puzzle} transaction and whoever fulfills the condition (provide the value h) will obtain 1 coin. The Tumbler publishes on the Blockchain a T_{solve} transaction which fulfills the condition and the Tumbler earns 1 coin. The client will obtain the value h and he will compute the solution.

But how can the client be sure that the encryption of the cipher text produces the correct solution of the puzzle in order to give the Tumbler the signature that is needed to withdraw the escrowed Bitcoin? The solution to this problem is given by the Cut and Choose method.

At the beginning, the client creates m real puzzles by blinding the puzzle which he received from the recipient m times and each time with a different blinded factor. After, the client creates n fake puzzles and encrypts the solution with the Tumblers' public key and blinds the puzzle. We must make clear that he already knows the solutions to the fake puzzles. Finally, the client shuffles the two kinds of puzzles and sends them to the Tumbler.

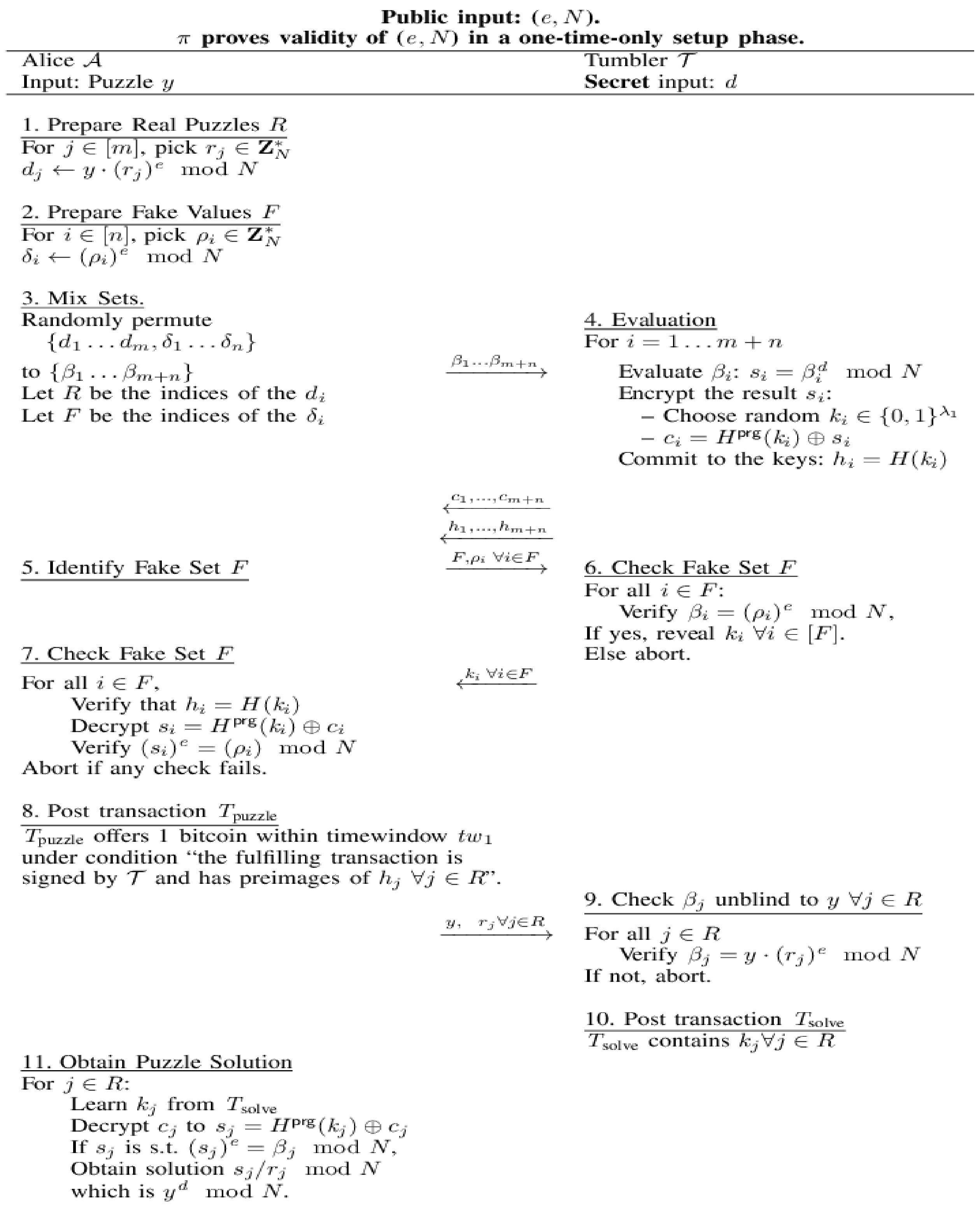
When the Tumbler receives the puzzles, he cannot distinguish the real from the fakes. So, the Tumbler now will solve all the puzzles and will return to the client m + n pairs of (c, h). The client reveals to the Tumbler the fake puzzles and asks their solutions. The Tumbler will return to the client the keys in order to decrypt the solutions. This won't bother the Tumbler because the client already knows the solutions.

By the time the client receives the keys, he verifies that all the fake puzzles were correctly computed. Then the client publishes a signed T_{puzzle} transaction in the Blockchain offering 1 coin in exchange for the key that will reveal the solutions to the real puzzles. Before the Tumbler reveals the solution, the client must prove that all the m real puzzles correspond to the original one.

So, the client sends the blinding factors of the real puzzles to the Tumbler in order to prove the above. The Tumbler after the verification he publishes on the Blockchain a T_{solve} transaction which contains the solution. Finally, the client gets the solution and the Tumbler the coin.

The above description provides a deep understanding to the protocol which is used by the TumbleBit. But the TumbleBit implements a small change in order to minimize the on-Blockchain movement to off-Blockchain. There must be some modifications so that the transactions T_{puzzle} and T_{solve} are implemented in an off-Blockchain transaction. The changes are taken place after the client has verified that the solutions of the fake puzzles are computed correctly

In more detail, the client instead of publishing the puzzle transaction, he signs it and sends it directly to the Tumbler. Also, the Tumbler after the verification instead of publishing at the Blockchain a T_{solve} transaction which contains the solutions, it sends it directly to the client. In figure 30 the RSA-Puzzle-Solver-Protocol algorithm is depicted



Public input: (e, N).
π proves validity of (e, N) in a one-time-only setup phase.

Alice $\mathcal{A}$		Tumbler $\mathcal{T}$
Input: Puzzle y		**Secret** input: d

1. Prepare Real Puzzles R
For $j \in [m]$, pick $r_j \in \mathbf{Z}_N^*$
$d_j \leftarrow y \cdot (r_j)^e \mod N$

2. Prepare Fake Values F
For $i \in [n]$, pick $\rho_i \in \mathbf{Z}_N^*$
$\delta_i \leftarrow (\rho_i)^e \mod N$

3. Mix Sets.
Randomly permute
$\{d_1 \dots d_m, \delta_1 \dots \delta_n\}$
to $\{\beta_1 \dots \beta_{m+n}\}$
Let R be the indices of the d_i
Let F be the indices of the δ_i

$\xrightarrow{\beta_1 \dots \beta_{m+n}}$

4. Evaluation
For $i = 1 \dots m+n$
Evaluate β_i: $s_i = \beta_i^d \mod N$
Encrypt the result s_i:
– Choose random $k_i \in \{0,1\}^{\lambda_1}$
– $c_i = H^{\mathsf{prg}}(k_i) \oplus s_i$
Commit to the keys: $h_i = H(k_i)$

$\xleftarrow{c_1, \dots, c_{m+n}}$

$\xleftarrow{h_1, \dots, h_{m+n}}$

5. Identify Fake Set F

$\xrightarrow{F, \rho_i \ \forall i \in F}$

6. Check Fake Set F
For all $i \in F$:
Verify $\beta_i = (\rho_i)^e \mod N$,
If yes, reveal $k_i \ \forall i \in [F]$.
Else abort.

$\xleftarrow{k_i \ \forall i \in F}$

7. Check Fake Set F
For all $i \in F$,
Verify that $h_i = H(k_i)$
Decrypt $s_i = H^{\mathsf{prg}}(k_i) \oplus c_i$
Verify $(s_i)^e = (\rho_i) \mod N$
Abort if any check fails.

8. Post transaction T_{puzzle}
T_{puzzle} offers 1 bitcoin within timewindow tw_1 under condition "the fulfilling transaction is signed by $\mathcal{T}$ and has preimages of $h_j \ \forall j \in R$".

$\xrightarrow{y, \ r_j \forall j \in R}$

9. Check β_j unblind to $y \ \forall j \in R$
For all $j \in R$
Verify $\beta_j = y \cdot (r_j)^e \mod N$
If not, abort.

10. Post transaction T_{solve}
T_{solve} contains $k_j \forall j \in R$

11. Obtain Puzzle Solution
For $j \in R$:
Learn k_j from T_{solve}
Decrypt c_j to $s_j = H^{\mathsf{prg}}(k_j) \oplus c_j$
If s_j is s.t. $(s_j)^e = \beta_j \mod N$,
Obtain solution $s_j / r_j \mod N$
which is $y^d \mod N$.

Figure 30:RSA-Puzzle-Solver-Protocol[44]

3.2.3.2 Puzzle-Promise Protocol

The Puzzle-Promise-Protocol is used in the interaction between the recipient and the Tumbler in order to create a puzzle. Let's assume in our case that the recipient seeks the solution and for exchange it will gain 1 Bitcoin from the Tumbler.

The main goal in this protocol is the Tumbler to provide to the recipient a puzzle-promise-pair (c, z).The promise c has been encrypted with the solution of the puzzle z. The promise makes sure that if the recipient finds the solution to the puzzle, he will obtain the Tumblers' signature in order to withdraw the escrowed coin

But how can the recipient be sure that the encryption of the cipher text unlocks the correct signature in order to withdraw the escrowed Bitcoin? The solution to this problem is given by Cut and Choose method

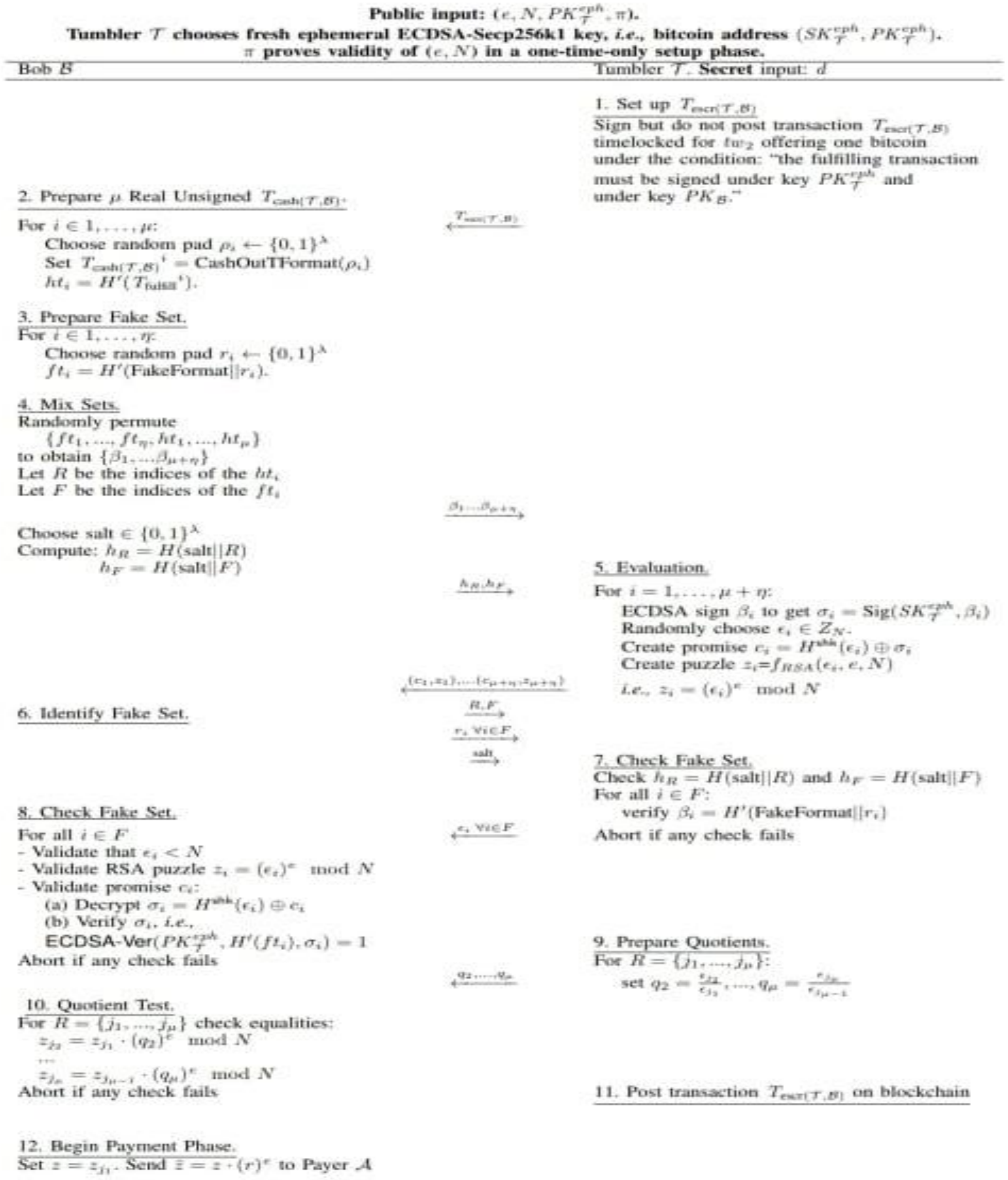
In more detail, at the beginning the recipient creates m real unsigned cash transactions. After, the client creates n fake transactions and hashes them as well as the real transactions. Then he shuffles the transactions and sends them to the Tumbler. The Tumbler now must evaluate the transactions that are sent and for each transaction the Tumbler returns at the client a puzzle promise pair (c,z).

The recipient at first checks the pairs that correspond to the fake transactions. This is done by verifying that the pairs are correctly formed. In order to achieve this, he must know the solutions to the puzzles in fake pairs. So the recipient provides the needed evidence to the Tumbler that the n pairs are fake. When this is done, the Tumbler in exchange offers him the fake pairs. If the verification is completed with no problems from the client, it is verified that there is a small probability that the Tumbler will misbehave.

With the cut and choose method the recipient can be sure that at least one of the real pairs are correctly formed but how can the recipient have the knowledge of which puzzle is correctly formed? The solution here is given by the quotient-chain technique. In more detail by solving the puzzle z_1, the recipient is able to solve all the other puzzles that were formed.

Finally, we must underline that in the case that the recipient obtains more than one signatures when opening a real pair, this doesn't cause any problem. This happens because it is predefined that the recipient will obtain only one coin in exchange for the signature. Also, this signature is encrypted under temporary key. So the recipient cannot withdraw more coins.

In figure 31 the Puzzle-Promise-Protocol algorithm is depicted.



Public input: $(e, N, PK_{\mathcal{T}}^{eph}, \pi)$.
Tumbler $\mathcal{T}$ chooses fresh ephemeral ECDSA-Secp256k1 key, *i.e.*, bitcoin address $(SK_{\mathcal{T}}^{eph}, PK_{\mathcal{T}}^{eph})$.
π proves validity of (e, N) in a one-time-only setup phase.

Bob $\mathcal{B}$ | Tumbler $\mathcal{T}$. **Secret** input: d

```
Tumbler:
1. Set up $T_{escr(\mathcal{T},\mathcal{B})}$
Sign but do not post transaction $T_{escr(\mathcal{T},\mathcal{B})}$
timelocked for $tw_2$ offering one bitcoin
under the condition: "the fulfilling transaction
must be signed under key $PK_{\mathcal{T}}^{eph}$ and
under key $PK_{\mathcal{B}}$."

    $\xleftarrow{T_{escr(\mathcal{T},\mathcal{B})}}$

Bob:
2. Prepare $\mu$ Real Unsigned $T_{cash(\mathcal{T},\mathcal{B})}$.
For $i \in 1, \dots, \mu$:
    Choose random pad $\rho_i \leftarrow \{0,1\}^\lambda$
    Set $T_{cash(\mathcal{T},\mathcal{B})}{}^i$ = CashOutTFormat($\rho_i$)
    $ht_i = H'(T_{fulfill}{}^i)$.

3. Prepare Fake Set.
For $i \in 1, \dots, \eta$:
    Choose random pad $r_i \leftarrow \{0,1\}^\lambda$
    $ft_i = H'(\text{FakeFormat}||r_i)$.

4. Mix Sets.
Randomly permute
    $\{ft_1, \dots, ft_\eta, ht_1, \dots, ht_\mu\}$
to obtain $\{\beta_1, \dots \beta_{\mu+\eta}\}$
Let $R$ be the indices of the $ht_i$
Let $F$ be the indices of the $ft_i$

    $\xrightarrow{\beta_1 \dots \beta_{\mu+\eta}}$

Choose salt $\in \{0,1\}^\lambda$
Compute: $h_R = H(\text{salt}||R)$
         $h_F = H(\text{salt}||F)$

    $\xrightarrow{h_R, h_F}$

Tumbler:
5. Evaluation.
For $i = 1, \dots, \mu + \eta$:
    ECDSA sign $\beta_i$ to get $\sigma_i = \text{Sig}(SK_{\mathcal{T}}^{eph}, \beta_i)$
    Randomly choose $\epsilon_i \in Z_N$.
    Create promise $c_i = H^{shk}(\epsilon_i) \oplus \sigma_i$
    Create puzzle $z_i = f_{RSA}(\epsilon_i, e, N)$
    i.e., $z_i = (\epsilon_i)^e \mod N$

    $\xleftarrow{(c_1, z_1), \dots (c_{\mu+\eta}, z_{\mu+\eta})}$

Bob:
6. Identify Fake Set.

    $\xrightarrow{R, F}$
    $\xrightarrow{r_i \forall i \in F}$
    $\xrightarrow{salt}$

Tumbler:
7. Check Fake Set.
Check $h_R = H(\text{salt}||R)$ and $h_F = H(\text{salt}||F)$
For all $i \in F$:
    verify $\beta_i = H'(\text{FakeFormat}||r_i)$
Abort if any check fails

    $\xleftarrow{\epsilon_i \forall i \in F}$

Bob:
8. Check Fake Set.
For all $i \in F$
- Validate that $\epsilon_i < N$
- Validate RSA puzzle $z_i = (\epsilon_i)^e \mod N$
- Validate promise $c_i$:
    (a) Decrypt $\sigma_i = H^{shk}(\epsilon_i) \oplus c_i$
    (b) Verify $\sigma_i$, i.e.,
    ECDSA-Ver$(PK_{\mathcal{T}}^{eph}, H'(ft_i), \sigma_i) = 1$
Abort if any check fails

Tumbler:
9. Prepare Quotients.
For $R = \{j_1, \dots, j_\mu\}$:
    set $q_2 = \frac{\epsilon_{j_2}}{\epsilon_{j_1}}, \dots, q_\mu = \frac{\epsilon_{j_\mu}}{\epsilon_{j_{\mu-1}}}$

    $\xleftarrow{q_2, \dots, q_\mu}$

Bob:
10. Quotient Test.
For $R = \{j_1, \dots, j_\mu\}$ check equalities:
    $z_{j_2} = z_{j_1} \cdot (q_2)^e \mod N$
    ...
    $z_{j_\mu} = z_{j_{\mu-1}} \cdot (q_\mu)^e \mod N$
Abort if any check fails

Tumbler:
11. Post transaction $T_{escr(\mathcal{T},\mathcal{B})}$ on blockchain

Bob:
12. Begin Payment Phase.
Set $z = z_{j_1}$. Send $\bar{z} = z \cdot (r)^e$ to Payer $\mathcal{A}$
```

Figure 31: Puzzle-Promise Protocol[44]

3.2.3.3 Core Protocol

In the starting phase the recipient requests from the Tumbler to open a payment channel. In order to achieve this the Tumbler escrows one Bitcoin by publishing in the Blockchain a 2-of-2 escrow transaction which is symbolized $T_{T,B}$. The recipient must fulfill the transaction condition, so he can be able to withdraw the coins. The condition is in fact his as well as the Tumblers' signature verified by their public keys. The escrow transaction has a time window, which means that if the recipient hasn't obtained timely the signatures, the Tumbler can take back the escrowed coins.

On the other side, the Tumbler requests at the client to open a payment channel. In order to achieve this, the client escrows one Bitcoin by publishing in the Blockchain a 2-of-2 escrow transaction which is symbolized $T_{C,T}$. The Tumbler must fulfill the transaction condition so he can be able to withdraw the coins. The condition is in fact his as well as the clients' signature verified by their public keys. It is important that the escrow transaction has a time window which means that if the Tumbler hasn't obtained by that time the signatures, then the client can take back the escrowed coins. It must be mentioned that the time window among the Tumbler and the recipient is larger than the time window among the client and the recipient.

The interaction between the Tumbler and the recipient starts with an off-Blockchain cryptographic protocol which is the puzzle-promise protocol. As mentioned, the Tumbler sends to the recipient a puzzle z which guarantees the recipient that if he finds the solution he will obtain 1 Bitcoin (in the reality the signature that is needed for the escrow transaction). The puzzle z that the Tumbler creates is an RSA encryption with a solution ε. Finally he will encrypt the signature that is needed to unlock the escrow transaction under the solution ε.The cipher-text c that is going to be created is then sent to the recipient.

When the payment channels are opened and the puzzle is sent to the recipient, everything is complete in order to proceed to the next phase. So, the recipient chooses a blinding factor and blinds the puzzle and creates puzzle z*.The blinded puzzle will be sent to the client which he must solve. After, the interaction between the Tumbler and the client starts with an off-Blockchain cryptographic protocol which is the puzzle-solving protocol. The target of the client is to get the blinded solution and send it to the Tumbler. The Tumbler needs the client's signature in order to unlock the escrows transaction condition and obtain 1 Bitcoin. So this puzzle assures that as long as both parties act in an honest way, both of them will be satisfied. This happens because the puzzle-solving protocol is a fair exchange protocol.

The blinded puzzle is solved by the Tumbler and returns the blinded solution to the client. Simultaneously the client provides at the Tumbler his signature. After, the client sends the blind puzzle to the recipient. And at the same time the Tumbler posts a fulfill transaction to the Blockchain with the two signatures that are demanded by the escrow transaction and withdraws the coins.

The recipient unblinds the solution and sends it to the Tumbler and decrypts the Tumblers' signature. Finally, the recipient posts a fulfill transaction to the Blockchain with the two signatures that are demanded by the escrow transaction and withdraws the coins.

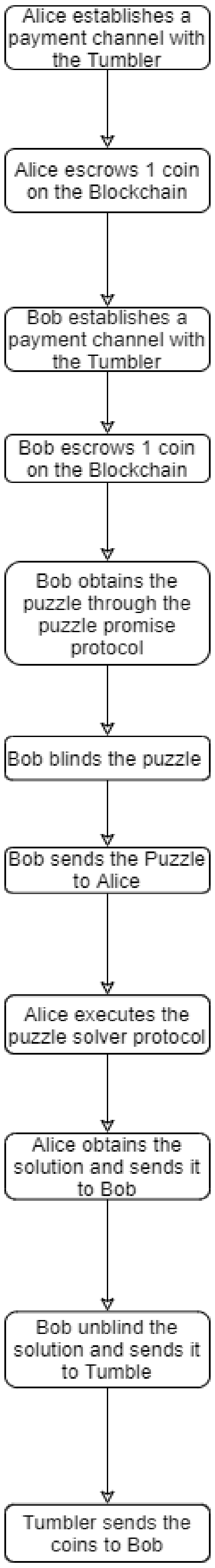

Alice establishes a payment channel with the Tumbler
Alice escrows 1 coin on the Blockchain
Bob establishes a payment channel with the Tumbler
Bob escrows 1 coin on the Blockchain
Bob obtains the puzzle through the puzzle promise protocol
Bob blinds the puzzle
Bob sends the Puzzle to Alice
Alice executes the puzzle solver protocol
Alice obtains the solution and sends it to Bob
Bob unblind the solution and sends it to Tumble
Tumbler sends the coins to Bob

Figure 32: TumbleBit Protocol

3.2.3.4 Generalization

TumbleBit came in order to achieve full anonimity as well as security,meaning no party participating in the protocol can steal coins.This technique is founded on a centralized service but uses a secure two-party computation and zero-knowledge proof so it can protect the peer's anonimity and safety.In cocnclusion,it was designed to upgrade the BlindCoin protocol which could not offer guarantees concerning the fairness of the protocol.But it could only provide accountability

3.2.3.5 Advantages

- The linkability, the sender's and recipient's anonymity is protected due to three variables. The first one is that all payments are implemented using k-anonymity, so it not possible to violate anonymity by comparing the values of the transactions. Secondly, the TumbleBit protocol functions based on phases and epochs. In more detail, all escrow transactions are published simultaneously when the escrow phase takes place. Also every escrow transaction transaction is cashed out when the cash out phase takes place. So during the Payment Phase all the payments from the sender to the recipient take place but those two entities never communicate directly. This results in blocking time attacks where the Tumbler increases or decreases its communication with the client targeting to distinct patterns of the recipients behavior. Finally, the anonymity due to the blinding of the puzzle that is needed to be solved
- The TumbleBit protocol provides theft prevention. Specially, due to the fact that the protocol uses 2-of-2 escrow transactions and puzzle solving, it assures that any entity gets paid, if the conditions of the agreement are met. Also the two puzzle protocols from their nature assure that the operation will be done in a legitimate and correct way
- The protocol is able to defend against DoS attacks. All transactions include a transaction fee that is destined for the miners who verify transactions. Due to the fact that the Tumbler does not trust the sender or the recipient it will not pay the fees when the escrow phase takes place. When the sender opens a payment channel with the Tumbler, the sender will not only take care of the escrowed coins but also of the transaction fee. In the same direction, when the Tumblers open a payment channel with the recipient, the transaction fee is going to charge the recipient. All the above will make DoS attacks very expensive so, it assumed that an attacker will not proceed with those attacks
- The protocol resists to Sybil attacks based on the same explanation that was provided for the DoS attacks

3.2.3.6 Disadvanatges

- The hash-locks require a lot of transaction space, which will result to additional Blockchain storage, network bandwidth as well as transaction fee
- The cut and choose protocol needs a heavy execution time, which is not able to satisfy the real-time application demands

3.3 Decentralized Mixing

3.3.1 CoinJoin

The basic idea of CoinJoin is "When you desire to implement a transaction, seek for someone else who also desires to implement a transaction and generate a joint transaction together". For instance, let's assume we have two transactions:, where one is from peer A to peer C and another is from peer B to peer D. These two independent transactions can be united together into one CoinJoin transaction, while inputs and outputs are not modified. The resulting joint transaction mixes the connection among inputs and outputs, thus the exact direction of data flow will be maintained secret to the other peers.

3.3.1.1 Core Protocol

CoinJoin is a decentralized mixing protocol in which all peers that are participating are obligated to sign a joint transaction. The peers maintain their anonymity and their funds cannot be stolen, because a peer will offer his signature only if all of the other peers reach an agreement concerning the transaction. If one peer denies signing the single transaction, then the protocol is aborted. In figure 33 the summary of CoinJoin protocol is depicted

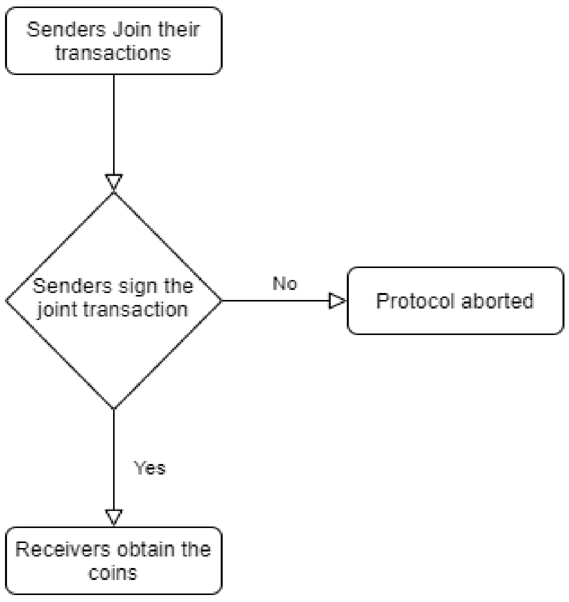

Senders Join their transactions
Senders sign the joint transaction
No
Protocol aborted
Yes
Receivers obtain the coins

Figure 33: CoinJoin Protocol

3.3.1.2 Generalization

In order for peers to face the DoS attacked that are applied in the centralized services, a decentralized mixing method is designed to permit a group of mutually untrusted peers to broadcast their messages at the same anonymously and without the existence of a third-party mixer. One more major advantage of this concept is the elimination of the need for mixing fees. So, for the above needs the CoinJoin protocol was described.

3.3.1.3 Advantages

- The joint transaction which will be formed will break the link among inputs and outputs so the anonymity of the direction that the data will go remains anonymous
- In the CoinJoin protocol, if a peer attempts to cheat in any way then the other members of the mixing won't provide their signature due to the break of the thief prevention protocol, so the transaction will not be implemented
- CoinJoin doesn't need additional mixing fees to implement the procedure because the mixing is being completed by the peers and not by a third central authority.
- CoinJoin doesn't need additional mixing fees to implement the procedure because the mixing is being completed by the peers.

3.3.1.4 Disadvantages

- The peers that participate in the mixing process will have knowledge of the details concerning the joint transaction due to the fact that the protocol doesn't provide internal unlikability
- The mixing protocol is vulnerable in DoS attacks. Consider that a malicious peer can produce a great number of independent virtual entities which are related with specific IP addresses as well as public as public keys. This type of attack has the ability to disrupt the mixing procedure by denying to sing the joint transaction
- A practical problem the CoinJoin presents is that the total number of peers participating in the mixing operation is not unlimited. This happens due to the fact that the protocol becomes more vulnerable to DoS attempts as well as the expotentional cost augmentation of the communication overhead. On the other side, the limitation on the number of participants decreases the anonymity of the protocol.

3.3.2 CoinShuffle

To assure the verifiability, the CoinShuffle protocol is created on the same paradigm as CoinJoin. A set of peers jointly produce a single mixing transaction and every one of them can independently assure that she will not lose coins by executing the transaction. In the scenario of a malicious attempt, the peer will not accept to sign the transaction.

But unlikability and robustness are the primary issues. In order to produce a transaction and at the same time providing guarantees that the input addresses cannot be connected to the output addresses peers shuffle their output addresses in an oblivious way. The shuffling method is founded on the accountable anonymous set messaging protocol Dissent. Also, the shuffling offers robustness in the meaning that attacks that target to disrupt the protocol can be found by honest users. The malicious peers that attempted the attack can be recognized and disqualified from the process

3.3.2.1 Core Protocol

The entities that participate in CoinShuffle mixing protocol are exclusively the peers. The two techniques that are used are the k-anonymity and the multisignature**.** The final payments are implanted as one payment like it takes place in the CoinJoin payment

In the beginning, the announcement phase takes place where all the peers who are going to participate in the mixing process produce an ephermal encryption and decryption key. Then, they broadcast their public encryption key in the network. Also, every participant generates a new Bitcoin address which is formed in order to be his output address in the mixing process. The next phase is the shuffling process, where the output addresses of all peers are shuffled as it happens in a decryption mix network.

In particular, every peer (let’s assume peer i in a default shuffling order) utilizes the encryption key of each peer j>i in order to make a layered encryption of his output address. After, the peers execute a consecutive shuffling, beginning with the first peer. Every peer waits to obtain i-1 ciphertexts from peer i-1. When he receives it, the peer strips one layer of the encrypted data from the ciphertexts. Next, the peer appends his own ciphertext and randomly shuffles the resulting group. Then the peer transfers the shuffled group of ciphertexts to peer i+1. The same procedure keeps going on until the set of ciphertexts arrive at the last participant. Assuming that every peer behaves according to the rules of the protocol, the decryption that is implemented from the last peer generates a shuffled list of output addresses. Finally, the list is broadcasted into the network from the last peer.

Furthermore, the list that is published in the network must by verified by all peers that have participated in the shuffling. This is done from every peer independently by making sure that their output address is being placed in the output list. If all peers have verified the accuracy of the list, then each peer generates a transaction in order to send coins from all input addresses to the shuffled output addresses in the list. Then the peers sign their transaction with their private key and publish their signature in the network. When each participant has received the signatures from all other peers, generates a complete signed version of the mixing transaction. This transaction is considered to be verified and can be submitted in the network.

In the above process we have assumed that all steps have been implemented according to the CoinShuffle protocol. However, there is always the case of having a violation of the protocol. In this case the blame phase is triggered. In more detail, if a peer misbehaves, then an honest peer will be forced to inform the blame phase which will attempt to recognize the malicious peer. The malicious peer will then be disqualified from a next mixing round. The scenarios are three in which the blame phase starts.

In the first scenario, a participant doesn't have enough money at his wallet to implement the mixing or he has spent the coins needed before the mixing has reached his end. In this case, the network offers the proofs needed for the misbehavior. The second scenario is when the shuffling has been executed wrongly. In this case, the peers publish their decryption keys together with the messages they have obtained. This evidence permits to every peer to replay the computations of the remaining peers and identify the malicious one. The third scenario is when a peer sends a fake public key to the other peers during the announcement phase. Peers among them exchange messages in order to be sure that none peer has misbehaved. If after the exchange they have reached in the conclusion that there is a breach in the protocol, then the blame phase starts and due to the fact that all messages are signed by their owner, the malicious peer can be recognized. Specifically, the existence of two different messages that are owned by the same owner and the same broadcast is the proof needed to incriminate him. In figure 34 the summary of CoinShuffle protocol is depicted

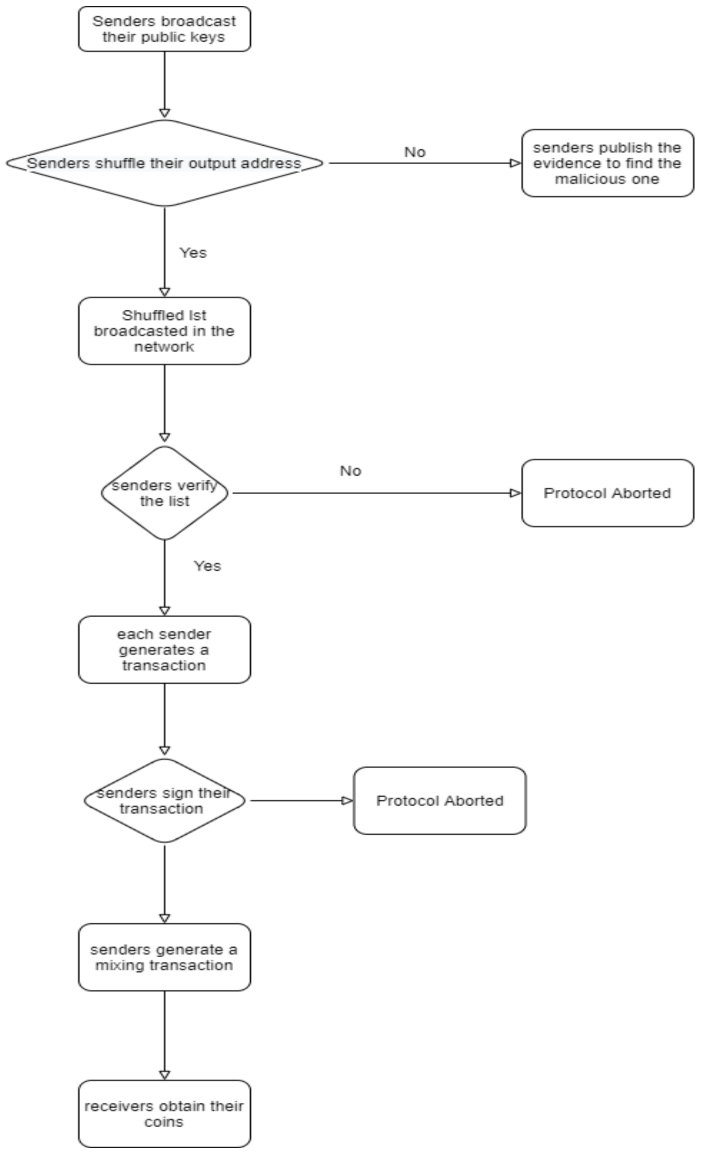

Senders broadcast their public keys
Senders shuffle their output address
No
senders publish the evidence to find the malicious one
Yes
Shuffled lst broadcasted in the network
senders verify the list
No
Protocol Aborted
Yes
each sender generates a transaction
senders sign their transaction
Protocol Aborted
senders generate a mixing transaction
receivers obtain their coins

Figure 34: CoinShuffle Protocol

3.3.2.2 Generalization

CoinShuffle has designed following the CoinJoin blueprints and it wiped out CoinJoin's main issue. In more detail, CoinJoin lacks internal unlinkability. In other words the peers taking part in the mixing process know information concerning the joint transaction. This information contains the destinations of transactions with which clients addresses are matched. This results in augmenting the probabilities of a Sybil attack as the number of peers in the network gets larger. To provide internal unlinkability to peers, CoinShuffle uses an anonymous set communication protocol to hide the peer's identifiers from one another. This technique achieves the internal unlinkability by using just layered encryption and shuffling

3.3.2.3 Advantages

- In every possible scenario that we pick to take, the linkability, the sender's anonymity and recipient's anonymity is protected due to three variables. The first one is to use k-anonimity by which it is not possible the possesors of the transactions by observing the flow of coins in the Blockchain. The second is the way that the mixing is implemented through the shuffling. The third is that multiple transactions are implemented through one transaction
- The CoinShuffle provides with theft prevention services through the blame phase in every case. In the case that the sender doesn't have enough coins to complete the transaction, the network provides the needed information and the protocol is aborted. Also, if the peer doesn't perform the shuffling correctly than other peers publish their description keys along with the messages exchanged in the intial phase of the protocol and it is aborted
- CoinShuffle doesn't charge the peers with additional mixing fees due to the decentralized nature that it has

3.3.2.4 Disadvantages

- The CoinJoin protocol presents a weakness concerning DoS attacks.This happens due to multi-signature technique, where every peer participating in the process, must sign at the shuffling phase the output list which has been published and at the final phase the transaction that is going to be implemented. So, if a peer is malicious he won't provide his signature and the protocol is going to be aborted
- CoinShuffle doesn't charge the peers with additional mixing fees due to the decentralized nature that it has

3.3.3 Xim Protocol

The protocol [45] functions in two stages: the discovery of a mix peer, and then the fair exchange with that peer. It also needed peers to use an arbitrary pool P where all peers in a specific pool synchronize the start and end of a mix round utilizing information distinguishable in the Blockchain. Xim protocol operates in a high level as described below.

Alice desires to mix an amount of coins that are placed in an address A. To start the process, Alice advertises a transaction that informs the peers in the network her desire to mix with another partner by tipping τ/2 coin from A to the miners. It must be underlined that she connects to the network through Tor network in order to keep her IP address anonymous. After other peers (also connected through Tor network) that want to get involved in a mixing process with Alice contact her. The peer (let’s assume he is named Bob) that is picked by Alice tips τ coin to the miners. Now the exchange among the two peers is verified by Alice by tipping again τ /2 coins to the miners. Following, δ coins are sent from A to B (an address which belongs to Bob) as well as δ coins are sent from B to A_1 (an address owned by Alice).The exchange of coins is executed in one logical step utilizing the Fair Exchange protocol. By the time Alice completed the mixing round she starts a new one. We assume that Alice wants to mix m*δ coins, so she will need more than one mixing round. If she wants, the m times that are required can be executed in the same time.

The advertising protocol is created in order for the Advertiser (Alice) and the Respondent (Bob) to spend τ on an advertisement. These recurring costs consist of an effective defense mechanism against Sybil attacks. Furthermore, due to the fact that they are paying to enter the pool of peers rather than for actual mixing, they assure that DoS attacks take place in a non-zero cost. Despite the fact that the advertisements are public, no peer can provide evidence that the two parties are connected. However, if respondent aborts the protocol before paying τ, the advertiser has the ability to reutilize her advertisement without paying new amount of τ/2 coins. In the same way, if A aborts the protocol after R pays τ, then R provides evidence that he committed to working with A. By the time the respondent has published the evidence in the network, the other peers will ignore the A advertisement and this will lead in to forcing her to make a new advertisement along with spending a new amount of coins.

3.3.3.1 Core Protocol

Initially, a peer that wants to mix his coins randomly picks to be the advertiser or the respondent. More specifically, an if statement is executed that checks if rand(0, 1) > 0.5. The peer is an advertiser with an address A and location $α_A$ if it returns true or he is an respondent with an address R and location $α_R$ if it returns false. Let’s assume that the peer is an advertiser named Alice. The locations are the addresses where the peers are receiving messages as the protocol executes.

After, Alice will advertise her desire to mix her coins by spending τ/2 coins as well, by informing other peers for the location $α_A$ that they can contact her. The messages that are destined for her will be encrypted with the public key of the address from which she published the advertisement.

So, several candidate respondents will come in touch with her by sending an encrypted message to the indicated address. The message consists of the N_a (nonce that uniquely recognizes the advertisement) they are answering to, their own nonce N_r and an address $α_R$ in which they will receive the reply in order to set up the fair exchange.

Then Alice will choose one of the respondents (let's assume that the peer picked is named Bob) in order to complete the mixing. She commits to his request by sending to location $α_A$, a signed message which contains N_a and the hash of N_r. The rest of the candidates that wanted to be Alice mixing partners will abort the protocol and they will attempt to find other advertisers. By the time Bob notices the commitment he must put his own response advertisement on the Blockchain. This will have a cost of τ + f (balance cost).The responder is assured that the cost of his advertisement corresponds to only Alice advertisement due to the fact that the message is encrypted with her public key. Finally, if both communicating parties behave honestly, than Alice will broadcast a response advertisement on the Blockchain costing t/2 coins that indicates publically that she is matching with a peer obfuscated as $h(N_r)$

There is also the case where a failure in the protocol appears. If Bob's advertisement is not put on the Blockchain by an indicated time, then A can cancel the matching and re-utilize her advertisement with no additional cost. On the other side, if Alice does not broadcast a response by an indicated time, then R provides evidence that Alice misbehaved by saving the below values to $α_A$ and $α_R$:

- the id of Alice's advertisement
- N_a the pairing message h(Nr)
- his knowledge of the true id N_r.

Every third entity has the ability to encrypt N_a and N_r, and pair them to Bob's advertisement for confirmation.

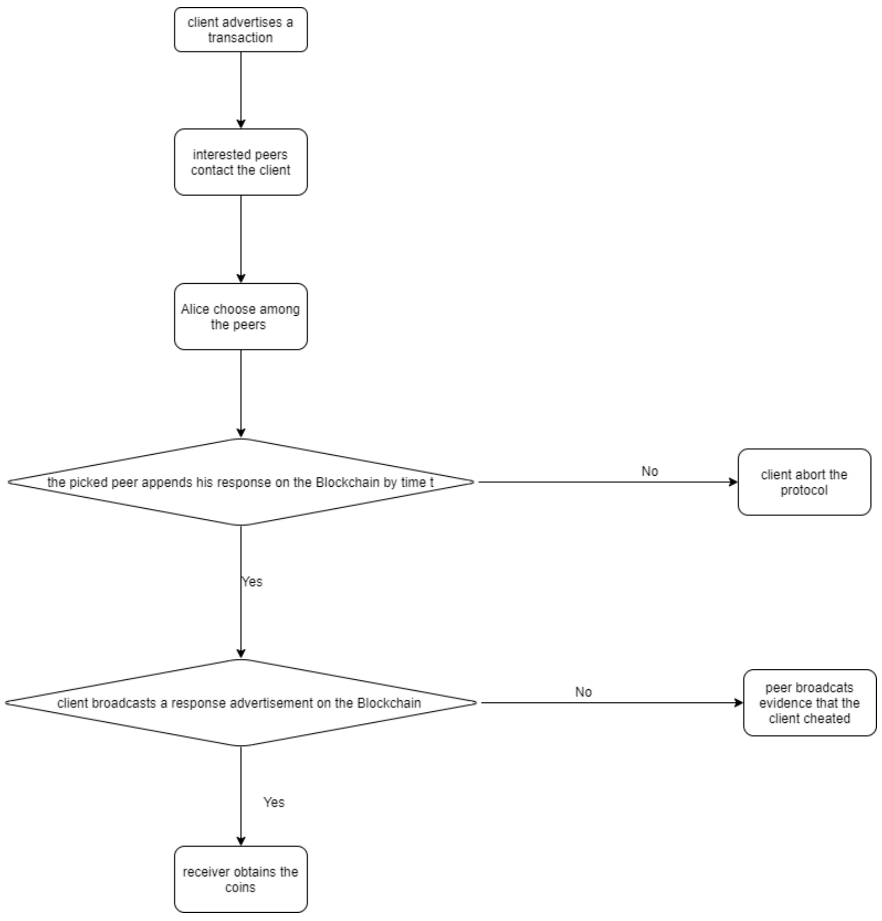

client advertises a transaction
interested peers contact the client
Alice choose among the peers
the picked peer appends his response on the Blockchain by time t
No
client abort the protocol
Yes
client broadcasts a response advertisement on the Blockchain
No
peer broadcats evidence that the client cheated
Yes
receiver obtains the coins

Figure 35: XIM Protocol

3.3.3.2 Mixing fees

If Alice aims to mix δ coin, then when she accesses the network she must have stored in her public address an amount of δ + n*τ + 5nf coins. The n*τ coin are needed in order to be able to afford the advertisement cost τ for every of the n mixing rounds. All transactions that are created demand a mining fee f that is paid to the Blockchain. Peers each creates four or five transactions during a mixing round accordingly on whether they behave as a respondent or a advertiser correspondingly. So, 5nf must be withhold as mining expenses. The network is responsible of determining the amount of transactions fees. In the contrary the tipping amount τ is defined by peers and it is as smaller as possible. Finally δ variable has the smallest value in order to be able to attract the maximum number of peers

3.3.3.3 Advantages

- Due to its interaction structure, Xim can maintain big anonymity sets, similar to the ones that are being assured by centralized mixers, assuming many peers are participating in the mixing process of the protocol
- Xim protocol offers protection against Sybil and DoS attacks
- Theft prevention is being offered in peers through the advertisements and the Blockchain where each entity can abort the protocol, if the other entities cheat

3.3.3.4 Disadvantages

- A large scale of the interaction takes place sequentially over the chain itself, so the waiting time for transactions to be gathered by miners, put inside the blocks, appended to the chain and substantially confirmed(by extending the chain) will take a lot of hours.

3.4 Smart Contract Mixing Protocols

3.4.1 Mobious Protocol

Initially, in this mixing protocol [46], in order to for Bob and Alice to be able to generate stealth keys, they share a secret, a nonce as well as Bob's master key. Each time Alice wants to trigger a transaction with Bob, she will utilize the shared secret in order to generate a fresh public key from Bob's master key. In case she attempted to send the coins directly to stealth address that Bob owned, the transaction could reveal the connection between them. This would happen even if the addresses used were never utilized before. So, the sender to able to overcome this problem she sends the amount of coins as well as the derived stealth key to the contract which is responsible for the mixing process. This leads that if Alice aims to transfer an amount that is larger than the input that the contract allows she is forced to spilt the amount in multiple mixing rounds.

When a sufficient number of senders have transferred in the contract coins in order to participate in the mixing, the stealth keys of the peers that are stored in the contract are utilized in order to create a ring. Now Bob is able to withdraw his money anonymously by generating a ring signature and ephermal public key to receive the coins

3.4.1.1 Stealth Adresses

Many cryptocurrencies are founded on stealth addresses [47] (or stealth keys). They represent public addresses that have been produced from a master key. One master key can produce a lot of different stealth keys but if the value that implemented the production is not known, then it is impossible for a peer to link the different addresses among the different peers.

In this technique, except the well known key generation algorithm, we must stipulate the algorithms that correlate with stealth addresses. The first one is the stealth public key algorithm. It is symbolized as spk ← SA.PubDerive(mpk,secret,nonce). In more detail, one stealth public key can be generated by giving the master key as input to the algorithm, the shared secret and a nonce. The second one is the stealth secret key algorithm and it is symbolized as ssk ← SA.PrivDerive(mpk,secret,nonce).

3.4.1.2 Algorithms Used In Mobious

As mentioned above the first interaction is the initialization of communication. During this event the client and the recipient, posses a secret key and they communicate in order to create a common basis for the next transactions. The following algorithms are used in this part of the Möbius protocol:

- tx ←− Deposit(sk_A, pk_B, aux): The client runs this algorithm to transfer a specific amount of coins into the smart contract.
- 0/1 ← VerifyDeposit(tx): The smart contract runs this algorithm in order to verify that the clients transaction is valid.
- ProcessDeposit(tx): If the transaction is confirmed, the smart contract executes this algorithm to refresh its internal state accordingly.

The second interaction between the peers is when the client is ready to inform the recipient that the agreed amount of coins have been sent to the smart contract and he can withdraw them. The following algorithms are used in this part of the Möbius protocol:

- tx←− Withdraw(sk_B, aux): The recipient runs this algorithm to withdraw his coins from the smart contract.
- 0/1 ← VerifyWithdraw(tx): The smart contract runs this algorithm to verify that the recipient's withdrawal is valid.
- ProcessWithdraw(tx): If the withdrawal is confirmed, the smart contract uses this algorithm to refresh its internal state accordingly.

In this mixing model the peers can take the roles of either senders or recipients. The basic role of the sender is to transfer the agreed amount of coins to the recipient through the smart contract.

The basic target of Möbius is to replace the on-Blockchain communication with off-Blockchain communication that is needed in order for the sender to communicate with the recipient. For this reason we assume that there are only two interactions among them. These are the initialization of their interaction and the other one is when the client informs the recipient that the coins are waiting to be withdrawn from the smart contract. Each one has an interaction with the smart contract. Finally in all the above communications we consider that the public state of the smart contract is provided as input.

3.4.1.3 Core Protocol

The first step in the protocol is the initialization of the smart contract which is placed in the address $id_{contract}$. The fields of the contract that must be specified are the following:

- participants: the multitude of peers that participate in the formation of the ring
- amt: the amount of coins that the contract will mix
- pubkeys[]: the public keys which are going to be used to form the ring
- senders[]: address of the peers that will participate in the mixing procedure
- sigs: the signatures seen until now which will be the verification tool for double withdrawal tries.

Adding to the code that is needed in order to append the above fields in the smart contract, there is also a code used to confirm deposit and withdrawal transactions. So there is no need to keep on-chain record of all signatures that where utilized in the past transactions and the contracts storage is erased after the last withdrawal is implemented.

The second step in the protocol is the initializations of the client's and the recipient's first interaction. The client must have knowledge of the master public key of the recipient so he can have the ability to send him coins. This knowledge can be obtained from an on-chain public key directory or it may be known from previous transactions. Then, the only action they need to do is to share the secret key and give nonce the value 0. If this directory doesn't exist, the initialization interaction is responsible to implement share of the muster public keys. The master public keys' role is to share the secret with the help of elliptic curve Diffie-Hellman (ECDH). Other tools can be also used for the sharing of the secret key if they fulfill the limitation that only the client and the recipient must know the secret.

The next step is the client to transfer the agreed amount of coins in the smart contract. In order for the client to be able to do the transfer, he must derive a new stealth key for the recipient. Then, he is able to create a transaction, through which he will transfer the stealth public key and the coins to the smart contract. The client is also obliged to publish this transaction.

By the time the transaction has reached the contract, it must be proved that is a valid transaction. This will be verified if the transaction is correctly formed, include the agreed amount of coins and the stealth public key is correct. The public key has a connection with a secret key that will be used by the recipient to withdraw the coins. The algorithm that will be used for the verification is VerifyDeposit. Assuming that everything goes well, the smart contract will renew the list with the keys it maintains by adding the one included in the transaction. The algorithm which will be used for this process is the ProcessDeposit. By the time the demanded numbers of peers have joined, the contract publishes a notification which will be used by the client. Then the client notifies the recipient that the coins are in the contract and sends him the address $id_{contract.}$

There are some cases where the protocol is not executed as we wish until this stage. The first case is that there is a chance that a time limit has been exceeded and the defined number of peers have not joined the contract. Two things may happen. Either the smart contract, by having access to the senders address, will return the coins back to their legitimate owner, or, the mixing process continues with the number of peers that have been included in the contract. We must underline that in the first case the availability is reduced and in the second one the anonymity is reduced. The path that the contract will follow will be defined by the creator of the contract.

Finally, we have reached the final phase of the Möbius protocol where the recipient is going to withdraw the coins form the smart contract. The first action for the recipient is to fetch from the pubkeys[] field the public keys which will be used to shape the ring. After, he will derive the stealth secret key that is related to the stealth public key of the recipient. This stealth secret key is the one that will give the ability to create a ring signature and withdraw his money from the smart contract into a ephermal address. This will be done using the algorithm Withdraw.

The contract from its side must verify that the withdraw transaction fulfills three conditions. The transaction is obliged to include a legitimate signature for the ring, the ring signature cannot be connected to any other signature used before in order to get coins and the ring has been made in a valid way. This will be done using the algorithm VerifyWithdraw. If the verification by the contract is successful, the contract will keep for the record the signature for probable next transactions and will form a transaction transferring the money to the ephermal address of the recipient. By the time all the peers have obtained their coins from the smart contract, it will erase all the fields except the peers that participate and the agreed amount of money that can be mixed. So the contract will be ready to start a next mixing phase.

As mentioned, before there is a probability that a limit time goes through and the recipient has not implanted the withdrawal transactions yet. In this case, there are two choices. The first one is not to put a time limit and allow the recipients to have an infinite time to make the withdrawal. The second is the return of the contract to its initial state and obtain any excess. The decision is once more on the creator of the contract. In figure 36 the summary of Möbius protocol is depicted.

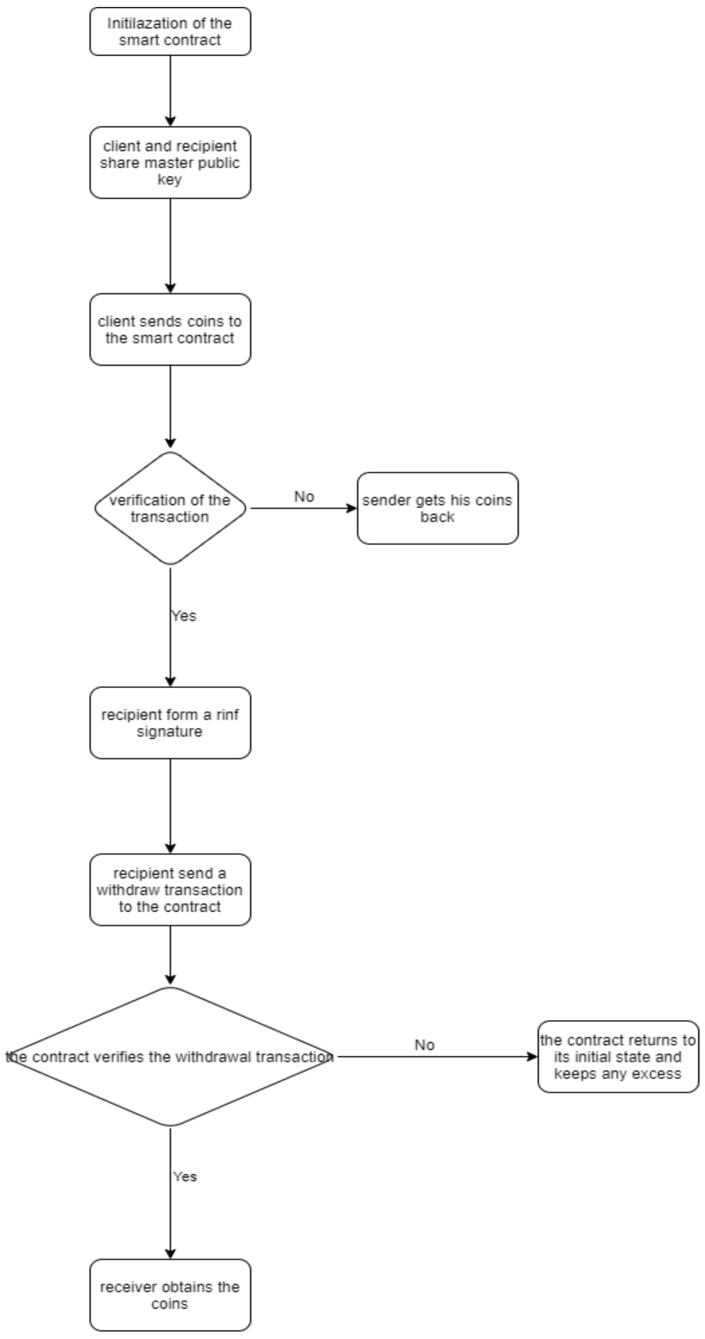

Initilazation of the smart contract
client and recipient share master public key
client sends coins to the smart contract
verification of the transaction
No
sender gets his coins back
Yes
recipient form a rinf signature
recipient send a withdraw transaction to the contract
the contract verifies the withdrawal transaction
No
the contract returns to its initial state and keeps any excess
Yes
receiver obtains the coins

Figure 36: Mobious Protocoll

3.4.1.4 Generalization

Despite the fact that there are many mixing protocols designed, the majority of them still suffer from disadvantages. The majority of decentralized mixing protocols demands a high level of coordination and permits each sender to learn the connections among senders and recipients. In centralized mixing protocols the whole process is based on the availability of the mixer. If the mixer goes offline or declines to transfer the money to the indicated recipient the whole protocol will be collapsed. In order to solve the above limitations the Mobious protocol was designed.

3.4.1.5 Advantages

- Mobious Protocol is able to resist to denial of service attacks
- Due to the stealth addresses and the ring signature the recipient anonymity is maintained
- If we evaluate the linkability from the view of an outsider or a sender, the linkability is maintained
- It achieves to replace a big part of the on-Blockchain communication with off-Blockchain communication
- Due to the implementation of the ring signature method, there is no way that a peer can steal coins. So, thr protocol provides theft prevention

3.4.1.6 Disadvantages

- The degree of anonymity depends on the number of peers participating in the mixing procedure. More peers mean that there is a smaller probability of a peer to guess the connection among inputs and outputs.
- The sender's anonymity is not always maintained
- If we evaluate the linkability from the view of recipient, it is violated

Conclusion

In conclusion, as the years pass and technology advances it is almost inevitable that electronic transactions will dominate over traditional transactions. This will also result to replacing physical coins with digital coins as well. So, it necessary to develop electronic transactions platforms that will not only be safe for every user, it will also provide anonymity.

As we examined in the thesis, there have been many attempts to enhance electronic transactions in the cryptocurrencies scheme. The first attempt, as mentioned in Chapter 2 was anonymous networks. We examined in depth how they function, their fundamental structure as well as their weaknesses. Anonymous networks have the ability to amplify the needs of the user but are not in position to solve the problem, which mainly is the anonymity

The mixing protocols came as the most promising solution to the issue. In the last chapter we explained all the mixing categories as well as their subcategories. It is obvious that as mixing protocols evolved the users were able to confront the problem better. It must be underlined that nowadays mixing protocols and smart contracts are considered a very efficient combination in order to solve the anonymity problem (in has been given with details in the problem statement section in the first chapter)

Future researches will focus on creating new mixing protocols as well as new cryptocurrencies. The academic community has made great steps toward this direction.

Abbreviations – Acronyms

BV	Block Version
MTRH	Merkle Tree Root Hash
TCP/IP	Transmission Control Protocol/ Internet Protocol
PBH	Parent Block Hash
TC	Transaction Counter
P2P	Peer-to-Peer
PoW	Proof of Work
PoS	Proof of Stake
DPoS	Delegated Proof of Stake
UNL	Unique Node List
PBFT	Practical Byzantine Fault Tolerance
ETH	Ethereum
XMR	Monero
XRP	Ripple
OR	Onion Router
OP	Onion Proxies
HTTP	Hypertext Transfer Protocol
I2P	Invisible Internet Project
TTP	Third Trusty Party

REFERENCES

[1] M. E. Hellman, “An Overview of Public Key Cryptography,” IEEE Commun. Mag., vol. 16, no. 6, May 2002, pp. 42–49.
[2] A. Shamir. How to share a secret. Commun. ACM, 22(11):612–613, 1979
[3] F. Charlon, \Openchain." [Online]. Available: https://www.openchain.org/
[4] G. Greenspan, \MultiChain Private Blockchain," https://www.multichain.com/ download/MultiChain-White-Paper.pdf, 2015
[5] D. Chaum, Blind Signature System. Boston, MA: Springer US, 1984, pp. 153-153
[6] F. Zhang and K. Kim, \Id-based blind signature and ring signature from pairings," in Advances in Cryptology | ASIACRYPT 2002, Y. Zheng, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 533-547
[7] N. van Saberhagen, \Cryptonote," 2013. [Online]. Available: https://cryptonote.orgwhitepaper.pdf
[8] M. Raikwar, D. Gligoroski, and K. Kralevska, ‘‘SoK of used cryptography in blockchain,’’ Cryptol. ePrint Arch., Tech. Rep. 2019/735, Sep. 2019.
[9] O. Goldreich and Y. Oren, \De_nitions and properties of zero-knowledge proof systems," Journal of Cryptology, vol. 7, no. 1, pp. 1{32, Dec 1994. [Online]. Available: https://doi.org/10.1007/BF00195207
[10] I. Miers, C. Garman, M. Green, and A. D. Rubin, \Zerocoin: Anonymous distributed e-cash from bitcoin," in 2013 IEEE Symposium on Security and Privacy, May 2013, pp.397-411.
[11] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, \Zerocash: Decentralized Anonymous Payments from Bitcoin," in 2014 IEEE Symposium on Security and Privacy, May 2014, pp. 459-474.
[12] B. Preneel, \The state of cryptographic hash functions," in School organized by the European Educational Forum. Springer, 1998, pp. 158-182.
[13] P. Gallagher and A. Director, \Secure hash standard (SHS)," FIPS PUB, vol. 180, p. 183,1995.
[14] M. Conti, E. S. Kumar, C. Lal, S. Ruj, "A survey on security and privacy issues of bitcoin", *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416-3452, 4th Quart 2018.
[15] S. Gupta and M. Sadoghi, ‘‘Blockchain transaction processing,’’ in Encyclopedia of Big Data Technologies. 2019, pp. 366–376.
[16] R. C. Merkle, “A digital signature based on a conventional encryption function,” in Advances in Cryptology — CRYPTO ’87: Proceedings. Springer Berlin Heidelberg, 1988, pp. 369–378
[17] I.-C. Lin and T.-C. Liao, “A survey of blockchain security issues and challenges,” IJ Netw. Security, vol. 19, no. 5, pp. 653–659, 2017
[18] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, ‘‘A survey on privacy protection in blockchain system,’’ J. Netw. Comput. Appl., vol. 126, pp. 45–58, Jan. 2019.
[19] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, self-published paper, August 19
[20] N. community, Whitepaper:Nxt, http://nxtwiki.org/wiki/Whitepaper:Nxt
[21] A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A provably secure proof-of-stake blockchain protocol, in: Annual International Cryptology Conference, Springer, 2017, pp. 357–388.
[22] D. Larimer, Delegated proof-of-stake (dpos), Bitshare whitepaper
[23] M. Castro, B. Liskov, et al., Practical byzantine fault tolerance, in: OSDI, Vol. 99, 1999, pp. 173–186.
[24] M. C. K. Khalilov and A. Levi, “A survey on anonymity and privacy in Bitcoin-like digital cash systems,” IEEE Commun. Surveys Tuts., to be published
[25] V. Buterin, "A next-generation smart contract and decentralized application platform", *white paper*, 2014
[26] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical report. https://bitcoin.org/bitcoin.pdf (2008)
[27] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, 2014.
[28] Litecoin Project. 2017. Litecoin—Litecoin Wiki. Available online: https://litecoin.info/Litecoin (accessed on 2 February 2017).
[29] The Dash Network. 2017. What is Dash?—Official Documentation. Available online: https://dashpay.atlassian. net/wiki/pages/viewpage.action?pageId=1146914 (accessed on 2 February 2017).
[30] Monero. 2017. About Monero. Available online: https://getmonero.org/knowledge-base/about (accessed on 2 February 2017)

[31] Ripple. 2017. Welcome to Ripple. Available online: https://ripple.com/ (accessed on 2 February 2017).
Svetlana Sapuric, and Angelika Kokkinaki. 2014. Bitcoin is volatile! Isn't that right? In Business Information Systems Workshops, Lecture Notes in Business Information Processing. Berlin: Springer, pp. 255–65.

[32] Y. Sompolinsky and A. Zohar. Accelerating bitcoin's transaction processing fast money grows on trees. Not Chains, 2013.

[33] P. Fairley, "Blockchain world - feeding the blockchain beast if bitcoin ever does go mainstream, the electricity needed to sustain it will be enormous," IEEE Spectrum, vol. 54, no. 10, pp. 36–59, October 2017

[34] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in Financial Cryptography and Data Security: 17th International Conference, FC 2013. Springer Berlin Heidelberg, 2013, pp. 6–24

[35] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. 1998. Anonymous connections and onion routing. *IEEE J. Select. Areas Commun.* 16, 4 (1998), 482–494.

[36] The Tor Project. 2015b. Tor Metrics Portal: Network. https://metrics.torproject.org/torperf.html. (October2015). Accessed March 2015

[37] Øverlier, L., and Syverson, P. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposiumon Security and Privacy* (May 2006), IEEE CS

[38] Dingledine, R., and Mathewson, N. Tor path specification.

[39] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-Resource Routing Attacks Against Tor. In WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society, pages 11–20, New York, NY, USA, 2007. ACM.

[40] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao, "One cell is enough to break Tor's anonymity," in Proceedings of the Black Hat Technical Security Conference, Las Vegas, NV, USA, July 25-30, 2009,pp. 578–589.

[41] F. Astolfi, J. Kroese, and J. V. Oorschot. I2P— The Invisible Internet Project. Accessed: Feb. 17, 2017. [Online]. Available: http://mediatechnology.leiden.edu/images/uploads/docs/ wt2015_i2p.pdf

[42] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. IACR Cryptology ePrint Archive, 2014:77, 2014.

[43] Luke Valenta and Brendan Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In FC, 2015

[44] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg. TumbleBit: an untrusted Bitcoin-compatible anonymous payment hub. In Proceedings of NDSS 2017, 2017.

[45] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore. Sybil Resistant Mixing for Bitcoin. In WPES'14: Workshop on Privacy in the Electronic Society, 2014

[46] Sarah Meiklejohn and Rebekah Mercer. M¨obius: Trustless tumbling for transaction privacy. PoPETs, 2018(2):105–121, 2018.

[47] G. Gutoski and D. Stebila. Hierarchical deterministic bitcoin wallets that tolerate key leakage. In R. Bo¨hme and T. Okamoto, editors, FC 2015, volume 8975 of LNCS, pages 497–504, San Juan, Puerto Rico, Jan. 26–30, 2015. Springer, Heidelberg, Germany

[48] S. Zhang and J.-H. Lee, ''Analysis of the main consensus protocols of blockchain,'' ICT Express

[49] Ghosh A, Gupta S, Dua A, Kumar N. Security of Cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects. J Netw Comput Appl. 2020