MailCoin — Blockchain Technology for Emails

Martin Grottenthaler

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Secure Information Systems

in Hagenberg

im Mai 2017

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Contents

Abstract

The email system lacks functionality compared to the traditional postal system. This project extends the functionality of email by uniquely linking emails with cryptocurrency transactions. In doing so, various functionalities can be implemented. One possibility is the introduction of email stamps. Like postage stamps they increase the price per email, discouraging spammers from sending spam because it is too expensive. Another system allows non-optional receive confirmations for emails, similar to the service "Einschreiben" by the Austrian "Post AG". Through this system, the receiver must confirm the reception of an email, to be able to read it. These methods were implemented and then analyzed through self-assessment and third-party feedback. The anti-spam use case has strong requirements and there are too many problems in the current system for it to be successful. The receive confirmations built on this system have lower requirements and are more promising. The solutions developed in this thesis provide a good basis for user-oriented implementations and further research.

Kurzfassung

Das E-Mail-System bietet weniger Funktionalität als das traditionelle Postsystem. Dieses Projekt erweitert die Funktionalität des E-Mail-Systems indem E-Mails mit Kryptowährungstransaktionen verlinkt werden. Aufbauend auf diesem System kann verschiedene Funktionalität implementiert werden. Eine Möglichkeit sind E-Mail Marken. Ähnlich zu Briefmarken erhöhen sie die Kosten pro E-Mail. Das bringt Spammer davon ab Spam zu senden, weil es zu teuer wird. Ein anderes System ermöglicht nicht optionale Empfangsbestätigungen für E-Mails, ähnlich zum Service „Einschreiben“ der österreichischen „Post AG“. In diesem System muss der Empfänger den Empfang einer E-Mail bestätigen, bevor er sie lesen kann. Diese Systeme wurden implementiert und analysiert durch Eigenanalyse und Feedback durch Dritte. Die Anwendungsmöglichkeit zur Spamverhinderung hat hohe Anforderungen und im aktuellen System gibt es zu viele Probleme, um erfolgreich zu sein. Die Empfangsbestätigungen haben niedrigere Anforderungen und sind vielversprechender. Die Lösungen, die im Zuge dieser Arbeit entwickelt wurden bieten eine gute Basis für benutzerorientierte Implementierungen und weitere Forschung.

Chapter 1

Introduction

The importance of blockchain technology is rising rapidly and it is increasingly used in various areas. The decentralization, immutability, trustlessness and public availability of information can be used for numerous use cases. One of these use cases is the extension of emails with blockchain technology. Emails are uniquely linked to cryptocurrency transactions, which allows extending emails with cryptographically secured data in a blockchain.

Blockchain technology can be used to implement services for emails, which are also provided in the traditional postal system such as an anti-spam mechanism, where spam is reduced by charging money for sending mail. With blockchain technology, such a system can be decentralized, which means a third party is not necessary. The system that was developed to accomplish this is called MailCoin.

This thesis is about MailCoin. MailCoin is a system to link emails with cryptocurrency transactions. MailCoin is also a cryptocurrency based on Bitcoin. The parameters of the cryptocurrency are chosen to be an ideal basis for linking emails with transactions. Through this system, the functionality of email is extended.

1.1 Research question

This master thesis aims to answer the following research question:

> How is it possible to uniquely link emails with transactions of a cryptocurrency and how can an anti-spam system and a system for receive confirmations be implemented on top of that?

This research question consists of these sub-questions:

1. How can emails and cryptocurrency transactions be uniquely linked together?
2. What are the requirements for a blockchain used in this system?

3. How can the anti-spam system and the system for email receive confirmations be implemented?
4. What are the requirements for these use cases to work?
5. What are the advantages for users of the systems?
6. What are the disadvantages for users as opposed to not using these systems?
7. Are there other use cases that can be based on MailCoin?
8. How can the MailCoin cryptocurrency be successful?

1.2 Related work

There are already various systems implementing messaging in a blockchain. For example, Bitmessage[1], Cryptamail[2], and SwiftMail[3]. These systems are very similar to each other. They implement a system for messages in a blockchain: messages are saved in a blockchain, and therefore users of the system cannot communicate with users of traditional email, at least not without making use of another method, which connects these systems to the email system. This means that all these are alternatives to email and not extensions like MailCoin. MailCoin extends email in a way that allows email users to simply ignore the additional data.

1.2.1 Receive confirmations

One of the possible areas of applications for MailCoin is to implement a system for receive confirmations. Even though read and receive confirmations seem very similar, there is a crucial difference. Receivers could send a read confirmation without actually reading a message or not send a read confirmation, even though the message was read. So, accuracy of a read confirmation cannot be proven. As opposed to that, a receive confirmation can be done easier: it only confirms that a user is able to read a message. The system implemented through MailCoin offers provable receive confirmations instead of read confirmations.

Systems which provide read confirmations already exist, for example, services like Yesware[4] and mailtrack[5]. These services provide email tracking to their customers. They rely on a tracking pixel, a 1 x 1 pixel sized image, which is sent in the email body. When the receiver opens the email, lots of email clients automatically load remote content. The service owner monitors requests for the embedded image and if the image is requested, this signals

[1] https://bitmessage.org/wiki/Main_Page
[2] http://www.cryptamail.com/
[3] http://www.johnmcafeeswiftmail.com/faq
[4] http://www.yesware.com/
[5] https://mailtrack.io/en/

that the email was opened. In this scenario, a third party is needed. It would also be possible to implement the system on the sender's side, without the need for a third party. Another more pressing problem is that this system is not reliable. It depends entirely on the receiver's email client loading remote content. Even though many email clients automatically load remote content, it is not guaranteed that all do. Therefore, recipients who use email clients, that do not load images automatically, could receive and read these emails without notifying the sender. The entire system could also be seen as a privacy infringement, as receivers could not be made aware that their behavior is tracked.

There are two standardized approaches to this problem. One is the concept of Message Disposition Notifications. It is a system where a sender can request a read confirmation from the receiver upon reading the email [12]. However, this only works if the receiver's email client supports the feature. The recipient will be asked whether they want to send a read confirmation to the sender, yet they do not have to. In fact, they can receive and read an email without sending this confirmation. Thus, if the sender does not receive a read confirmation, it does not necessarily indicate that the receiver did not read the email. Another approach is Delivery Status Notifications. These are not sent by the receiver, but by the receiver's email server. They are not used to signal that the receiver read the email, but rather to indicate that it was delivered to their mailbox. A sender cannot be sure that the receiver's server implements this functionality and if the email server is not owned by the receiver there is still a third party that needs to be trusted [17]. A serious problem with both these approaches is that they are not mandatory. A receiver is able to read an email without sending any of this information.

With MailCoin, a system for receive confirmations can be implemented that does not rely on a third party. The most important feature of this system is that it is not optional. The receiver cannot read the email if they refuse to send a receive recipe. This concept is described in more detail in Chapter 6.

1.2.2 Anti-spam mechanism

MailCoin can also be used as an anti-spam mechanism. To accomplish this, MailCoin mimics a characteristic of the traditional postal system: sending mail costs money. It is expensive to send a lot of mail and in lots of cases it is not profitable to do so. On the other hand, sending emails is virtually free and, due to that, there is a lot of spam. There are many different approaches that try to reduce spam. One of these methods is to imitate the postal system and make the sender pay for sending emails. For example, by using a proof-of-work system or by paying money.

Microsoft's Penny Black [16] project looks into this issue. The project

induced a few publication. Bankable Postage for Network Services [1] is particularly interesting. It describes a system similar to Kerberos[6]. It is a system with a central server that gives tickets to clients who want to send emails. Senders can spend their tickets to send emails and recipients can validate the tickets of received emails with the server. Receivers can also refund tickets to honest senders, therefore honest users are less impaired when using the system. A similar method can be implemented with MailCoin, with the advantage that it can be implemented without the need of a trusted central server.

Another interesting approach is hashcash[7]. Hashcash is a proof-of-work algorithm, which is used in most cryptocurrencies in the block generation algorithm and, therefore, to generate coins. It can also be used to prevent email spam. This is done by appending hashcash tokens to emails. A hashcash token is generated by spending a lot of computing power but can be easily verified. Thus, sending emails gets slightly more expensive for senders. This should not be a problem for normal email users, as they usually only send a few emails. For spammers, who send a lot of emails, this has a bigger impact, because they must generate a hashcash token for every email and every receiver. MailCoin is based on Bitcoin and for this reason uses hashcash to create MailCoins. Therefore, a MailCoin is an abstraction of a hashcash token. MailCoin transactions can be bundled with emails, similar to hashcash tokens, which are sent with emails as well. This has the same effect, as an email with a hashcash token: computing power was spent to create that MailCoin. A big advantage is that MailCoins can be transferred and stored. Computing can be done in advance and the generated MailCoin can be transferred. Another advantage is that if a sender sends MailCoins to the receiver, the receiver can send these MailCoins back to the sender to indicate that the received email was not spam. If this is done, recipients of spam with an attached MailCoin transaction are paid for receiving spam and honest senders get their spent MailCoins back. This system is described further in Chapter 5.

[6] http://web.mit.edu/kerberos/
[7] http://www.hashcash.org/

Chapter 2

Basics of Bitcoin

MailCoin is a system for linking emails with cryptocurrency transactions, which is described in Chapter 3 and a cryptocurrency, which is described in Chapter 4. It is important to understand the basics of a blockchain based cryptocurrency to understand MailCoin. This Chapter explains the basics of Bitcoin, on which the MailCoin cryptocurrency is based.

2.1 Transactions

In Bitcoin, users do not have accounts that hold their balance; users have various key pairs that allow them to spend Bitcoins. There are programs, called wallets helping users manage their Bitcoins.[1] A wallet is a storage for all the key pairs a user owns. They normally also allow users to generate new key pairs and create transactions.

Assuming Bob is new to Bitcoin and Alice wants to send 10 Bitcoins to Bob, Bob downloads a wallet software. He launches the wallet software and downloads and verifies the entire Bitcoin blockchain (see Section 2.3). He then uses his wallet software to generate a new key pair. The wallet keeps both keys of the new key pair secret and calculates a hash of the newly generated public key. This is the Bitcoin address associated with this key pair. Bob now gives this address to Alice. She creates a transaction that sends 10 Bitcoins to Bob's address. Bob sees the transaction and can now control the Bitcoins that Alice just sent to him.

A Bitcoin transaction consists of inputs and outputs. Inputs define where the Bitcoins for the transaction come from, outputs define what happens with the Bitcoins. Inputs reference outputs of earlier transactions. For an input to be valid the creator of the transaction must prove that they are allowed to use these outputs. When the creator of a transaction creates an output, they specify the number of Bitcoins in this output and the conditions

[1] https://bitcoin.org/en/glossary/wallet

for using it in an input. The usage of an output as an input in another transaction is called "spending" this output.

Bitcoin transactions are not just used to send money from Alice to Bob. An output contains a script that specifies the conditions under which this output and, therefore the associated Bitcoins, can be spent. This script is called Pubkey Script[2]. To satisfy these conditions, somebody who wants to spend an output sets the script in the input in which they want to spend the output. This script is called the Signature Script[3]. The Signature Script contains data, needed for the Pubkey Script to succeed. When evaluating the transaction, the Signature Script and the Pubkey Script are concatenated. First, the Signature Script and then the Pubkey Script are executed. The Pubkey Script verifies the data provided in the Signature Script. After the script is executed, it should leave "true" on the stack indicating that it succeeded.

The most generic type of transaction output type is Pay-to-PubkeyHash (P2PKH)[4]. A transaction with a P2PKH output is basically a transaction sending Bitcoins from Alice to Bob. Alice specifies that whoever has the private key, belonging to the address defined in the Pubkey Script, can spend this output. Alice sends Bitcoins to the address provided by Bob. Even though P2PKH is commonly called a transaction type, it is actually an output type and is referred to as such in this thesis.

A private key and a public key are an ECDSA key pair [6]. A Bitcoin address is mainly a hash of the Bitcoin public key and a checksum. This adds an additional layer of security because normally the public keys are kept private and only Bitcoin addresses are published. Thus, even if there was a security problem with ECDSA, allowing attackers to calculate private keys from public keys, they would only know Bitcoin addresses and would still have to breach the hash functions to get the keys. If one of the layers was breached, the protocol could be updated to use another algorithm.

Bob now wants to send three of his Bitcoins to Charlie. To do so, Bob has to find the transaction that Alice created to send Bitcoins to him earlier. It is a P2PKH transaction. He wants to send Charlie three Bitcoins, to do that he creates a P2PKH transaction. He uses the output from the transaction with which Alice has sent him Bitcoins as the input for this new transaction and adds an output that sends three Bitcoins to Charlie's address.

The sum of the inputs' values minus the sum of the outputs' values cannot be negative. The margin between the two is the transaction fee. The transaction fee is paid for the transaction to be included in the blockchain. It is theoretically optional, but, especially if there are many transactions, transactions with a higher transaction fee are normally included in the block-

[2]https://bitcoin.org/en/glossary/pubkey-script
[3]https://bitcoin.org/en/glossary/signature-script
[4]https://bitcoin.org/en/glossary/p2pkh-address

chain faster (see Section 2.2). Using the example of Bob sending Bitcoins to Charlie, the input is an output with ten Bitcoins and the output to Charlie's address is three Bitcoins. In this case, the fee would be seven Bitcoins, which is unreasonable high. It is not possible to only spend parts of an output, however, Bob can send some of the Bitcoins back to himself. To do so, Bob creates a new key pair and adds an output to the transaction, sending six Bitcoins to his own newly generated address. This is called a change output.[5] Bob now has a transaction with one input of ten Bitcoins and two outputs, one sending three Bitcoins to Charlie and one output sending six Bitcoins back to himself. Now, the fee for this transaction is one Bitcoin.[6] For the transaction to be ready to be published in the network, Bob must sign the transaction. With his signature, Bob proves that he has the required private keys to spend the input. Bob needs to sign the transaction using the private key belonging to the address he gave to Alice in the first example. After the signature is added, he can publish the transaction in the Bitcoin network. If he did everything correctly, then Charlie will get the three Bitcoins and Bob still has six Bitcoins left.

In order for a signature to be verified, the public key is required. The signer not only generates and attaches the signature to the transaction but also their public key. This public key, and not only the associated address, is now publicly known and, therefore, one layer of security is lost. This address should not be used to receive Bitcoins anymore, because if ECDSA had a problem and it could be possible to generate private keys from public keys, then these Bitcoins could be spent by anyone. Due to this issue, Bitcoin addresses should not be reused, instead new ones should be created.

The Bitcoin network is a peer-to-peer network of Bitcoin nodes, which download and verify the whole transaction database. They verify transactions and relay them to other nodes. If a new transaction is verified correctly by a node, it is relayed to connected nodes, which also verify and relay the transaction. With this mechanism, a valid transaction is eventually propagated throughout the whole network. As soon as the transaction is saved in the blockchain, the transaction is confirmed. This process is described in detail in Section 2.3.

Another type of output is a Pay-to-ScriptHash (P2SH) output[7]. In a P2SH output, the script itself is not included, but a hash of the script, therefore keeping the transaction size low. To spend the output later, the spender needs to know the script that, when hashed, results in the hash defined in the Pubkey Script. The spender needs to provide the required data to satisfy the script and the script itself as the Signature Script. A spender needs to know the data to fulfil the script conditions as well as the

[5] https://bitcoin.org/en/glossary/change-address

[6] This is still an unreasonable high fee. It is used in this example because the explanation is easier with whole numbers.

[7] https://bitcoin.org/en/glossary/p2sh-address

full script. This script is called the redeem script. With P2SH outputs, any script can be defined as the Pubkey Script without having to worry about the transaction size. Only the spender has a higher transaction size, because they need to include the whole script.

One thing that can be implemented with P2SH transactions is multi-signature. A multi-signature output is an output that can only be spent when signed with multiple keys. When creating a multi-signature output, it is defined how many keys are needed to spend this output and how many keys there are in total. These parameters are also used to refer to the output. A 2-of-3 multi-signature output, for example, is an output where at least two of the three possible keys are needed to spend it. The hash of the multi-signature script is called a multi-signature address and users can send Bitcoins to this address through a transaction containing a P2SH output. These outputs can only be spent if the spending transaction contains the required number of signatures.

There are various areas of application for multi-signature transactions. Alice and Bob could use a 2-of-2 multi-signature address as their shared account. If one of them wants to spend this money, the other one must consent. A 2-of-3 multi-signature address could be used as an escrow solution. Alice wants to sell a product to Bob. Charlie is an independent third party. Alice, Bob, and Charlie create the multi-signature address with their public keys. Bob sends Bitcoins to the multi-signature address. These Bitcoins can then only be spent again if at least two out of the three persons agree. If everything is in order and Bob receives the product, Bob and Alice agree to send the Bitcoins to Alice. If, due to whatever issue, they cannot make a decision, Charlie has to determine the course of action is. He must side with one of them and cosign a transaction. For example, by giving the Bitcoin to Alice or Bob or solving the situation differently. None of them can do anything with the Bitcoins without another party giving their consent. In a traditional escrow system, Bob would send the money to Charlie who holds the money until Bob receives the product. In this system, Charlie must be trusted to not steal the money.

Another possibility for dealing with the Pubkey Script is to just put data in it. The “OP_RETURN” script opcode marks a transaction output as invalid. Therefore, this output cannot be spent anymore. Because everything after the “OP_RETURN” opcode is irrelevant to Bitcoin payments, arbitrary data can be added. This can be used to store data in the blockchain.

2.2 Block

Bitcoin transactions are saved in blocks. Blocks are chunks of data, consisting of a reference to the previous block (see Section 2.3) and a list of transactions. The actors, who create blocks are called miners and they nor-

mally use special dedicated hardware for the creating of blocks, as it takes a lot of computing power. This is because a valid block has the requirement that the value of the hash of this block is below a certain number. This is called the difficulty[8] and it is adjusted every 2016 blocks, depending on the current mean time between blocks. If the current mean time between blocks is higher than the target time, the difficulty is lowered, otherwise it is raised. The higher the difficulty, the harder it is to create a block. To create a block that has a hash with the required value, a nonce is added to the block data. This nonce is incremented until the hash of the block complies with the requirement. The number of SHA-256 hashes that need to be calculated before a valid block is found, depends on the difficulty, because the higher the difficulty, the lower the number of possible hashes that comply with the difficulty requirement. To give miners an incentive to create blocks, every created block gives the creator a block reward and the transaction fees of all transactions in the block. Via block rewards, new Bitcoins are created. Because the block rewards are halved approximately every 4 years, there is a maximum number of Bitcoins.

It is the miners' choice which transactions to include in a block. It makes sense for the miners to include all available transactions, because they can collect all the transaction fees. Currently, the maximum block size is 1 MB. If there are more transactions available than can fit into a new block, miners must decide which ones to include and which ones to leave out. It would make sense for them to include the transactions with the highest transaction fees. This means, that when there are more transactions than can fit into a block, the transaction fees also rise because there is a competition to get a transaction into a block. Whether this is a problem depends entirely on the main use case of a cryptocurrency. For a cryptocurrency that depends on lots of transactions with low value, this would be a big problem as the transaction fee should only be a fraction of the transaction value in order to to be viable. For MailCoin this would be a serious problem. A higher maximum block size or a lower time between blocks would allow for a higher throughput and should therefore lower the transaction fees per transaction again once more.

2.3 Blockchain

A blockchain is the basis for most cryptocurrencies, as is the case with Bitcoin. Like its name suggests, a blockchain consists of multiple blocks of data, linked together. Every block is linked to the previous block in order for the blockchain to be followed back to the first block from any given block in the chain. It is guaranteed that blocks are created chronologically, because the link to the previous block is the hash of the previous block. The hash

[8] https://bitcoin.org/en/glossary/difficulty

of a block cannot be known before it is generated. Blocks that already have a few blocks built on them are impractical to modify, as this would change the hash of all blocks built upon them and therefore all blocks would have to be created again. This would only work if miners with more than 50 % of the hashing power of the network worked together to reverse the blockchain, because then these miners could create more blocks in a shorter period of time than the rest of the miners and eventually create a longer blockchain. The attackers need more than 50 % of the total hashing power because the longer blockchain is the one considered valid.[9]

The first block is called the genesis block[10] and is hard coded; it is the anchor of trust of the whole blockchain. The distance between a block and the first block is called the block height. Because it is a chain and not a tree, there cannot be two blocks with the same height; every block has only one previous and at most one next block. This means that as soon as a new block is created, all miners start from the beginning, trying to create a new block referencing this block. As soon as one of the miners can produce a new block the cycle starts again. Honest miners usually do that, but they are not obligated to. If there are two blocks with the same height at the end of the blockchain, this is called a fork. Miners must decide on a block to continue building the blockchain on. This could happen naturally when two miners produce a block at similar times, but could also happen because an attacker wants to reverse transactions. The block on that the majority of miners build on will eventually result in the longer blockchain. The longer blockchain will be the accepted one. In these two scenarios, the shorter chain will most probably be discarded. There is another scenario though, which would potentially result in two valid blockchains. It could be in the cryptocurrency's interest to reverse transactions or change the rules for what a valid block is and what is not. If the clear majority of miners agreed to the changes, the result would be the same as described before, but if there was a considerable number of miners that want to keep the old version alive as well, both versions could live, therefore creating two versions of the blockchain. This would not be a big problem; users would just have to decide which path to take from the fork on, to use the desired version of the blockchain. Figure 2.1 depicts an example formation of a blockchain.

The blockchain holds every Bitcoin transaction ever made and therefore it holds the status of every Bitcoin ever created. This data is needed to verify transactions, because it needs to be verified whether the Bitcoins, that a user wants to move around, really exist.

Once a transaction is created and published in the network, it is relayed to all the nodes. This transaction has 0 confirmations right now because it was not yet included in a block. An attacker could create two conflicting

[9] https://bitcoin.org/en/glossary/51-percent-attack
[10] https://bitcoin.org/en/glossary/genesis-block

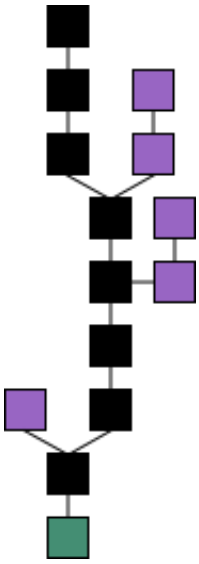


Figure 2.1: The green block is the genesis block. All other blocks are built onto this genesis block. The black blocks are part of the longest chain. The purple blocks were generated in a fork and were not built on by the majority of miners. Image source [25].

transactions and relay them; only one of them would eventually be included in a block, even though some nodes might have seen the transaction, that will eventually be dropped, first. Just because a transaction is seen, it does not mean that it will be included in a block. Thus, it is dangerous to accept transactions that are not yet included in a block. As soon as a transaction is included in a block, it has one confirmation. Accepting a transaction with one confirmation is much more secure, but as described before in this section, an attacker could create a fork and still reverse the transaction. Intentionally creating a fork is expensive and, therefore, it is adequate to accept transactions with one confirmation, as long as the transaction value is low. With every block built upon the block with the transaction, the number of confirmations of the transaction is incremented. For high-value transactions it would be a good idea to wait until there are a few confirmations before accepting a transaction, as with every confirmation it becomes computationally harder to reverse the transaction.

A blockchain can be private or public and permissioned or permissionless. The data in a public blockchain can be read by everyone; the data in a private blockchain can only be read by defined parties. When using a permissionless blockchain, everyone can create blocks, writing data in the blockchain. In a permissioned blockchain write access is restricted to certain participants.

Bitcoin uses a public, permissionless blockchain. Everybody can join and use the network and, furthermore, everybody can see all transactions and

data in the blockchain. Everyone who wants to participate can download and verify the blockchain. If somebody wants to download the blockchain they download it from another user of the network and verify it. They now act as a node of the network that accepts, verifies and relays transactions and lets other users download the blockchain as well. There is no central server from which the blockchain is downloaded; it is a peer-to-peer network. Every node of the network acts as client and as server. Users do not have to trust anybody. They just have to know the genesis block, which can be used to verify the blockchain downloaded from an untrusted source. To sum it up, one can say that the Bitcoin blockchain is an immutable database built on a trustless decentralized peer-to-peer network.

Because Bitcoin is a peer-to-peer network, when a node is started it cannot simply connect to a central server. It needs to find other nodes to connect to. It would not be feasible to brute force the IP address space until a server is found. The first time a node is started, it needs the IP address of at least one other node that is already connected to the peer-to-peer network. Bitcoin clients normally come with a hard coded list of known Bitcoin DNS seeds. A DNS seed is a DNS name which, when queried, returns a list of addresses of known Bitcoin nodes. The DNS seeds are maintained by Bitcoin community members. The list of addresses that can be queried can easily be updated by the DNS seed operator without changing the address of the DNS seed. The information gathered by these requests should not be blindly trusted, as it is not authenticated and malicious seed operators could supply a list of malicious nodes. This node information is used to connect to the network for the first time. The node can now learn the addresses of other nodes in the network by querying known nodes for the addresses of nodes they are connected to. With this mechanism, a node could learn the addresses of all nodes in the network. A node does not need the addresses of all nodes in the network, however, it saves the addresses of connected nodes. This allows connecting to the network through previously connected nodes and eliminates the need to rely on DNS seeds [6].

Chapter 3

Uniquely linking emails and blockchain transactions

MailCoin is a system for linking emails with cryptocurrency transactions. The use cases for this system are described in Chapter 5 and Chapter 6. Theoretically, every blockchain solution that meets certain requirements can be used as a basis. These requirements are described in the respective use cases. The basic requirements are described in Section 3.3. The MailCoin cryptocurrency, which is designed to perfectly meet these requirements, is described in Chapter 4.

As elaborated on in Chapter 1, hashcash is a big inspiration for MailCoin. In hashcash, there is already a model to create unique headers for emails. This model was analyzed and adapted for the use in MailCoin. In addition to the reference to the transaction that needs to be added to the email, a reference to the email needs to be affixed to the transactions as well. Both references are described in the next two sections. It should be noted, that this chapter describes the basic system. Some use cases might require additional information, as described in the detailed description of the use cases in Chapter 5 and Chapter 6.

3.1 Link in the email

Emails need to reference the corresponding transaction. This is done by appending a new email header. The new "MailCoin-TxID" header adds the transaction ID to the email. The receiver of an email can search the blockchain for the transaction with this ID. This is enough to reference the corresponding transaction, but, to verify the reference from the transaction back to the email, there needs to be information that can be referenced in the transaction. This information is included in the "MailCoin-Header". This header is inspired by the hashcash header and it is shown in Figure 3.1.

The first element of the MailCoin header is a version number. The version

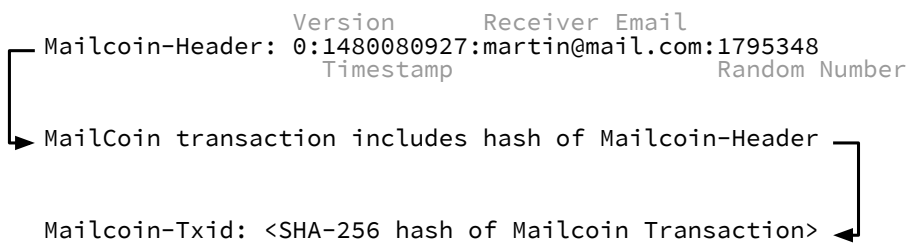


Figure 3.1: The connection between the MailCoin email headers and the transaction.

number is currently 0, but it could be updated if there were changes made to the format of this header. The second element is the current Unix time, indicating when the header is created. This can be verified by the receiver. The time stamp should not be in the future and it should also not be too far in the past, relative to the receiving time of the email. The next element is the recipient's email address. This, in combination with the Unix time stamp, is used to link the header to the email. The last element is a random number to make it even more unlikely that there are two identical stamps and it also makes the hash less predictable.

As shown in Figure 3.1 the "MailCoin-Header" is hashed and included in the transaction. An email's recipient now calculates the hash of the received "MailCoin-Header" and compares the hash with the value published in the transaction with the transaction ID in "MailCoin-TxID".

3.2 Link in the transaction

A MailCoin transaction follows the same rules as a Bitcoin transaction. For a transaction to reference an email, a MailCoin output needs to be added to the transaction. To create such an output the "MailCoin-Header" or at least its hash is required. With this hash, a MailCoin transaction output is created. The Pubkey script of this output is described in Figure 3.2. Besides the Pubkey script, an output also contains a MailCoin amount. Because of the "OP_RETURN" in the pubkey script, this output is unspendable [6]. It normally does not make sense to use an amount greater than 0 here, because the output could never be spent and therefore the MailCoins would be lost.

Just for the linking of emails to transactions, it is enough to have one MailCoin output. It should be noted, however, that the sum of the amounts of inputs minus the sum of the amounts of outputs is the transaction fee. The transaction fee cannot be negative and is the incentive for miners to include the transaction in a block; it should be high enough for the miners to include it, but also not too high, because that would be a waste of MailCoins. The

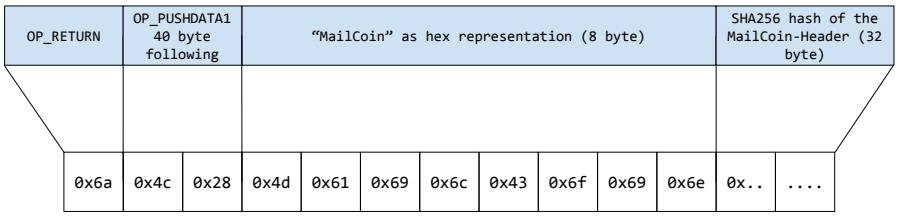


Figure 3.2: The connection between the MailCoin email headers and the transaction.

transaction fee should be adjusted based on the current network congestion and the size of the transaction. A transaction containing a MailCoin output should normally also include a change output.

Even though the linking as well as all the use cases are based on the MailCoin cryptocurrency described in Chapter 4, the system is designed to work with any Bitcoin-based cryptocurrency. In general, it depends on how much the users are willing to spend on transaction fees and how long they are willing to wait for a transaction to be confirmed.

3.3 Requirements

The system was developed to have low requirements and to be used with most cryptocurrencies. It needs to be possible to create a transaction with an unspendable output that only includes data. The transaction fees should be low because transaction fees are email fees. The transaction that is linked in the email has to be confirmed, so it cannot be reversed easily. The time it takes for a transaction to confirm effectively delays the sending of emails. To keep that delay time minimal, the cryptocurrency should also have a short mean time between blocks.

3.4 Transaction malleability

Transaction malleability is the problem that it is possible for unconfirmed transactions to be changed after they are signed without invalidating the signature. This is attainable because the signature does not sign the whole transaction; the Signature Script is not signed because this would mean the signature had to sign itself, which would not be possible. Therefore, any malicious node could add opcodes that change the Signature Script. Even though this cannot change the outcome of the script, because the script would not terminate successfully, this changes the transaction ID. This is a crucial issue to take into consideration when using MailCoin as it relies on

this transaction ID. To be completely sure, users would need to wait until their transaction is confirmed at least once, check if the transaction ID is still the same, and only then they could safely use their MailCoin transaction ID in their email.

In a real-world scenario, the problem is not as severe. It would only happen if a malicious node changed the Signature Script, other nodes accepted this malformed transaction and it was included in a block before the real transaction. The transaction is non-standard and, thus, honest nodes would not accept it. Furthermore, an attacker would not get anything out of this attack, because they could only change the transaction ID but nothing else.

Transaction malleability is a problem especially when transactions use unconfirmed transactions as input. Due to transaction malleability, these transactions could be invalid. This is a concern especially in the "Lightning Network" [22]. "Segregated Witness" could be implemented in the future to solve the transaction malleability problem and it would also bring other advantages [2]. "Segregated Witness" and the "Lightning Network" are described in detail in Chapter 8.

Chapter 4

MailCoin as a cryptocurrency

MailCoin is also the name of a cryptocurrency, which is based on Bitcoin, specifically on the implementation in Go[1] by btcsuite[2]. This implementation was adapted to the need of linking emails to cryptocurrency transactions, and therefore some parameters needed to be changed.

Besides changing parameters that control how the system works, some parameters needed to be changed to create a new blockchain. The most important one is the genesis block. The genesis block is the first block in a blockchain. It is hard coded and is the anchor of the whole blockchain. Every new block refers to the previous block and is therefore ultimately linked to the genesis block. It must be publicly known, as it is the trust anchor for the whole blockchain. Another important feature of a genesis block is that it should include information that proves that it was not generated before a certain date. This guarantees that the mining of the cryptocurrency did not start before this date. This is important because otherwise the creators of a cryptocurrency could mine lots of blocks and generate block rewards, for themselves, before releasing the code to the public. The Bitcoin genesis block, for example, includes the headline of "The Times" from January 3, 2009. The MailCoin genesis block includes the hash of the Bitcoin block with height 466500. This is the MailCoin genesis block:

```
1 // genesisCoinbaseTx is the coinbase transaction for the genesis blocks for
2 // the main network, regression test network, and test network (version 3).
3 var genesisCoinbaseTx = wire.MsgTx{
4   Version: 1,
5   TxIn: []*wire.TxIn{
6     {
7       PreviousOutPoint: wire.OutPoint{
8         Hash:  chainhash.Hash{},
9         Index: 0xffffffff,
10      },
11      SignatureScript: []byte{
```

[1] https://golang.org/

[2] https://github.com/btcsuite

```
12          0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* hash of */
13          0x00, 0xe8, 0xb7, 0xa1, 0xe5, 0x9c, 0x25, 0xf1, /* bitcoin */
14          0x62, 0x07, 0x6e, 0x2d, 0x5d, 0xbe, 0x79, 0x90, /* block  */
15          0x01, 0xe5, 0x0a, 0xde, 0xff, 0x46, 0x4c, 0x23, /* #466500 */
16        },
17        Sequence: 0xffffffff,
18      },
19    },
20    TxOut: []*wire.TxOut{
21      {
22        Value: 0x12a05f200,
23        PkScript: []byte{
24          0x41, 0x04, 0x67, 0x8a, 0xfd, 0xb0, 0xfe, 0x55, /* |A.g....U| */
25          0x48, 0x27, 0x19, 0x67, 0xf1, 0xa6, 0x71, 0x30, /* |H'.g..q0| */
26          0xb7, 0x10, 0x5c, 0xd6, 0xa8, 0x28, 0xe0, 0x39, /* |..\..(.9| */
27          0x09, 0xa6, 0x79, 0x62, 0xe0, 0xea, 0x1f, 0x61, /* |..yb...a| */
28          0xde, 0xb6, 0x49, 0xf6, 0xbc, 0x3f, 0x4c, 0xef, /* |..I..?L.| */
29          0x38, 0xc4, 0xf3, 0x55, 0x04, 0xe5, 0x1e, 0xc1, /* |8..U....| */
30          0x12, 0xde, 0x5c, 0x38, 0x4d, 0xf7, 0xba, 0x0b, /* |..\8M...| */
31          0x8d, 0x57, 0x8a, 0x4c, 0x70, 0x2b, 0x6b, 0xf1, /* |.W.Lp+k.| */
32          0x1d, 0x5f, 0xac, /* |._.| */
33        },
34      },
35    },
36    LockTime: 0,
37  }
38
39  // genesisBlock defines the genesis block of the block chain which serves as the
40  // public transaction ledger for the main network.
41  var genesisBlock = wire.MsgBlock{
42    Header: wire.BlockHeader{
43      Version:    1,
44      PrevBlock:  chainhash.Hash{},
45      MerkleRoot: genesisMerkleRoot,
46      Timestamp:  time.Unix(0x591955FB, 0), // 2017-05-15 07:17:15
47      Bits:       0x1d00ffff,
48      Nonce:      0,
49    },
50    Transactions: []*wire.MsgTx{&genesisCoinbaseTx},
51  }
52
53  // genesisHash is the hash of the first block in the block chain for the main
54  // network (genesis block).
55  var genesisHash = chainhash.Hash([chainhash.HashSize]byte{
56    0x68, 0xb9, 0xff, 0x66, 0x08, 0x4b, 0xf9, 0x29,
57    0x15, 0x5d, 0xa8, 0x00, 0xcd, 0x8f, 0xeb, 0xf1,
58    0x44, 0xd3, 0xfb, 0xb8, 0x56, 0xab, 0xa9, 0x4f,
59    0xbc, 0x32, 0x08, 0x8b, 0xa3, 0x71, 0xa8, 0xf0,
60  })
```

The Bitcoin client includes hard coded DNS seeds. Because MailCoin is based on a different blockchain, these DNS seeds are not valid anymore and were replaced by MailCoin DNS seeds. The Bitcoin client also included

Table 4.1: Parameters of MailCoin

	Bitcoin	*MailCoin*
Symbol	BTC	MLC
Coin limit	21 Millions	420 Millions
Proof-of-work algorithm	SHA-256	SHA-256
Mean block time	10 minutes	30 seconds
Difficulty retarget	14 Days / every 2016 blocks	14 days / every 40320 blocks
Block reward details	halved every 210000 blocks	halved every 4200000 blocks
Initial block reward	50 BTC	50 MLC
Max block size	1 MB	1 MB

a hard coded list of checkpoints. These were added in version 0.3.2 of the Bitcoin client as a security feature. They lock in the blockchain up to the checkpoints. Even if an attacker had more than 50 % of the hashing power of the total mining network, they could not reverse transactions made before this point [18]. These checkpoints had to be removed in MailCoin, because MailCoin is based on a different blockchain, but should be reintroduced at a later point, for the MailCoin blockchain, as this serves as an important security feature.

4.1 Parameters of MailCoin

Some parameters, which dictate the basic functionality of MailCoin, were changed to make it better suited for email usage. Table 4.1 gives an overview of the differences between MailCoin and Bitcoin.

The coin limit, the mean time between blocks and the block reward details are closely related. To allow faster transactions the mean block time was reduced by a factor of 20 to 30 seconds. That means the average time before a transaction is included in the blockchain is 30 seconds. 30 seconds seems like an acceptable waiting time because emails are not completely instant either. As pointed out by Vitalik Buterin in [5], certain problems arise when the time between blocks is too short. Therefore, the time between blocks was only reduced to 30 seconds. The shortened time between blocks also allows for more transactions in the same amount of time, because now a block, which can hold up to 1 MB transaction data is generated every 30 seconds and not just every 10 minutes.

In Bitcoin, the difficulty to generate new blocks is adapted every 2016 blocks or approximately every 14 days. In MailCoin the difficulty is adapted every 40320 blocks. Because of the lower mean time between blocks, this

also happens every 14 days.

The estimated time between block reward halving stays the same to keep the same controlled supply properties as in Bitcoin. Therefore, the number of generated blocks before the generation reward is halved is 20 times as high as in Bitcoin. The block reward is halved every 4,200,000 blocks. With a mean time of 30 seconds between blocks, the block generation reward is halved approximately every 4 years. The initial block reward stays the same, namely 50 MailCoins. Therefore the maximum amount of MailCoins is 420 million.

The used proof-of-work algorithm was not changed. It is still SHA-256. This is because there is not really a problem with SHA-256. Other cryptocurrencies changed the proof-of-work algorithm, because specialized hardware for Bitcoin mining became available and they wanted to give users the ability to mine with normal hardware again and, in doing so, keep the currency decentralized. However, eventually, specialized mining hardware was developed for these algorithms as well. The proof-of-work algorithm could be changed in the future, but currently, there is no reason to change the mining algorithm. It is not a problem that MailCoin can be mined with specialized hardware. Complete decentralization is not one of MailCoin's goals.

The maximum block size has not changed either. For MailCoin it is highly important to have low transaction fees, because transaction fees are basically email fees and, depending on the use case, there should not be a high fee for emails. If the transaction fees rise, because there are too many transactions and not enough space in the blocks to accommodate all transactions, actions should be taken to counteract this problem. The simple solution seems to be to increase or remove the maximum block size. Because blocks are generated 20 times as often in MailCoin as in Bitcoin, the throughput of the network should also be 20 times as high. This is not a problem at the moment, but could become one in the future. Bitcoin is currently also struggling with this problem, hence MailCoin can simply make use of the solution, which, hopefully, will have already been implemented by Bitcoin.

Besides these changes, MailCoin works just like Bitcoin. This allows complete compatibility with Bitcoin.

4.2 Source code

The implementation by btcsuite has some important distinctions from the official Bitcoin implementation. The most crucial difference is that the blockchain and the wallet are separate. There are two programs btcd[3], the blockchain, and btcwallet[4], the wallet. These were forked, renamed into mlcd and mlcwallet and adapted. This modular approach allows for adapting or even

[3] https://github.com/btcsuite/btcd

[4] https://github.com/btcsuite/btcwallet

replacing components easily without breaking the whole system.

The only component that was changed is the blockchain component. The changes described earlier in this chapter were applied. The architecture of btcd makes it easy to adapt. All information about the genesis block is defined in the file *genesis.go* and all parameters are defined in *params.go*.

The changes in the *genesis.go* file have already been explained earlier in this chapter. This section examines the parameter changes. These are the changes in the *params.go* file:

```
var MainNetParams = Params{
  ...
  DNSSeeds:    []string{
    ''162.252.172.67'',
  },
  ...
  SubsidyReductionInterval: 4200000,
  TargetTimespan:           time.Second * 30 * 40320,// 14 days
  TargetTimePerBlock:       time.Second * 30,        // 30 Seconds
  ...
  // Checkpoints ordered from oldest to newest.
  Checkpoints: nil,
  ...
}
```

The “DNSseeds” parameter is a hard coded list of DNS seeds. These are DNS servers that return a list of MailCoin nodes for initial peer discovery. Currently, there is only one other peer included in this list, as it is the only one existing at this point. This list should be extended as soon as the network has gotten bigger.

“SubsidyReductionInterval” controls after how many blocks generated, the block generation reward is halved. “TargetTimespan” refers to the target time in seconds, in which interval the difficulty for generating blocks should be adjusted and “TargetTimePerBlock” is the target mean time between the generation of blocks. These two values are used to determine the difficulty of the creation of blocks in MailCoin.

The “Checkpoints” parameter holds the hashes and height in the blockchain of blocks that are set as checkpoints. As there is currently no MailCoin blockchain, hashes cannot be set here yet. This variable should be updated to include regular checkpoints of the MailCoin blockchain at a later point.

Chapter 5

Anti-spam mechanism

Gudkova et al. found that approximately 57 % of all emails were spam in the second quarter of 2016 [11]. This project aims to reduce this number. MailCoin can be used as a mechanism for spam prevention by copying a property of the traditional postal system.

In the traditional postal system, users must pay to send letters. This is primarily because the process of delivering letters costs money. Email delivery, on the other hand, is almost free. Unwanted mail exists in both of these messaging systems, however, in the traditional postal system, senders have to pay much more to send messages. They must ask themselves whether it is really worth it to send mail that is most likely to be thrown away without generating any income for them. Senders cannot just send anything because they need a high percentage of receivers to, for example, buy something to break-even. In the email system, the percentage of needed buyers is much lower and because of that, the quality of the advertisement can be much lower as well. Rao and Reiley compare the prices and needed conversation rates to break-even of advertisement via the postal system, botnet spam and other ways of communication in [23]. According to [23] the cost per thousand impressions (CPM) is between 250 $ and 1000 $ in the traditional postal system. In comparison, with botnet spam, the cost per thousand impressions is between 0.03 $ and 0.05 $, depending on the used method. To break-even, the advertisers need a conversion rate between 2 % and 10 % when using the traditional postal system, but only a conversion rate between 0.00006 % and 0.0001 % when using botnet spam. This shows that email spam is very cheap in comparison to other forms of advertisement and it is not surprising that there is a lot of email spam because the needed conversion rate to break-even is very low. With MailCoin, a system can be implemented that allows adding a payment to emails. This system allows controlling the price per email, effectively controlling the break-even point for advertisers. The required conversion rate to still break-even, depending on required MailCoin transaction values, is shown in Figure 5.1. It was shown that even with

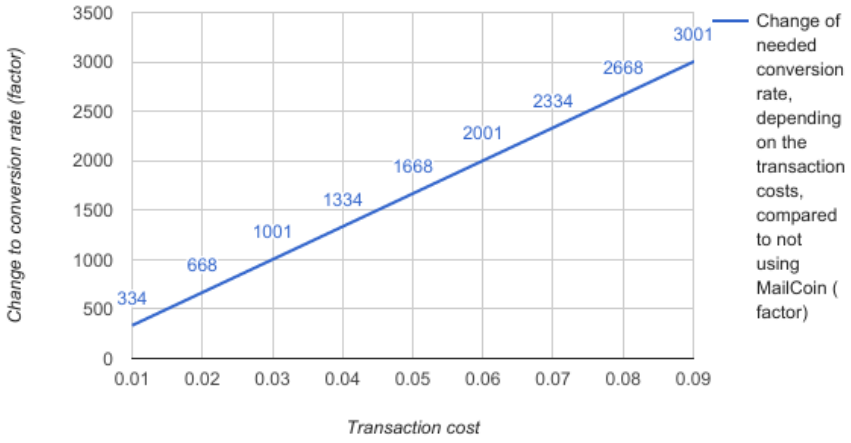


Figure 5.1: The change in the needed conversion rate compared to not using MailCoin. The y-axis represents the factor of how much higher the conversion rate needs to be, to break-even, in comparison to not using MailCoin. The x-axis represents the value of the MailCoin transaction.

MailCoin transactions, which have a value as low as 0.01$, the conversion rate needs to be 334 times as high to break-even. This shows that even a marginal email fee greatly affects the costs for spammers.

A big advantage of this system over the traditional postal system is that the beneficiary of the transaction can be the receivers of the emails instead of a third party. The receivers are paid for receiving emails and they can send the MailCoins back to honest senders. If receivers send the MailCoins back to honest senders, the real cost for honest communication partners is just the transaction fees. The transaction value are MailCoins used like a deposit that is paid back to honest senders. Therefore, the cost per mail could be high, because only dishonest senders would lose their deposit. Loder, Alstyne and Wash conclude in [15] that receivers do have an incentive to send back the received funds, because if they did not, people would stop communicating with them.

5.1 Email stamps

The MailCoin anti-spam system is a system similar to postage stamps, thus, the system is called email stamps. The sender attaches a stamp to an email. This stamp is attached in the form of an email header and it is the transaction ID of a transaction in the MailCoin blockchain. As in the traditional

postal system, these stamps need to be counterfeit-proof. To prove the validity, the receiver of an email with a stamp can check the referenced MailCoin transaction. It must have a reference back to the email. The link between the transaction and the email is described in Chapter 3. For email stamps, the transaction needs to include an output which gives MailCoins to the receiver of the email or some agreed upon third party, so that the email stamp has some value and the sender puts a certain number of MailCoins at stake. The validity of a stamp can be checked by the recipients themselves or anybody who can read the email headers and knows the MailCoin address of the receiver, for example, an external spam filter. The stamp itself is not reusable, because it is the transaction ID of a transaction, which uniquely links to a specific email. The transaction ID is a SHA-256 hash of the raw transaction and as long as there are no collisions found in SHA-256, the transaction ID identifies a unique, non-reversible transaction in the MailCoin blockchain. Although the stamp itself is not reusable, the outputs of the referenced MailCoin transaction, which are the sent MailCoins, can be reused by the receiver. The recipient could create a new email stamp, send the MailCoins back to the sender, use them in different way or just keep them for later usage.

A Sender who wants to use the system and wants to create email stamps for their email is required to own MailCoins. They can get them either by participating in any other protocol or by exchanging them for something else, like fiat currency or another cryptocurrency. They then have to ask the receiver for their email address, MailCoin address, and the minimum amount of MailCoins they need to receive to consider the email stamp as valid. The sender then writes the email like they usually do and generates the MailCoin email headers and MailCoin transaction, as described in Chapter 3. The MailCoin transaction additionally must include a payment output, to the receiver of the email. The sender then sends the email and publishes the transaction. They could alternatively wait until the transaction is confirmed in the blockchain to be sure the transaction hash cannot easily be changed after the transaction was published.

A receiver who uses the system must scan incoming emails for attached MailCoin transactions. If an email has a MailCoin transaction attached, the receiver is required to check the validity in the MailCoin blockchain. They verify that there is a valid link to a MailCoin transaction and gives MailCoins to the receiver or another trusted party. The receiver has to decide what to do with this information. A receiver could only accept emails with an email stamp or lower the spam score for emails with an email stamp. The receiver also needs to decide on the minimum number of MailCoins sent in a transaction to consider it a valid email stamp. In this thesis, if not stated otherwise, it is assumed that the receiver only accepts emails with a valid email stamp.

5.1.1 Communication between honest communication partners

This section describes the communication between the honest communication partners Alice and Bob. Both are using the MailCoin anti-spam system.

To send the first email, Alice needs to know Bob's email address, his MailCoin address, and the minimum amount of MailCoins that need to be sent with an email stamp to be classified valid by Bob. For Bob to reply to an email he needs the same information from Alice. They meet up face-to-face and exchange this information. Now both know each other's email addresses, each other's MailCoin addresses, and the minimum number of MailCoins to be sent. Meeting up face-to-face is not an appropriate solution. There should be a system to easily obtain that information. (see Chapter 8)

Alice writes an email to Bobs email address. She also attaches an email stamp to the email. The transaction referred to in the email stamp transfers MailCoins to Bob's address. Alice then publishes the transaction and sends the email. Bob's spam filter checks the incoming email. The spam filter sees the MailCoin headers. It checks the validity of the "MailCoin-Header" by comparing the information with the receiver email address and the receive time of the email. It then searches for the transaction with the transaction ID that is referenced in the MailCoin-TxID header. This transaction should contain a hash of the MailCoin-Header. The hash of the actual "MailCoin-Header" in the email and the hash in the transaction must be identical. After all these checks are completed successfully, the email is marked as "not spam" and forwarded to Bob's inbox. Bob can now read the email and check the new balance in his MailCoin wallet. After reading the email, Bob decides that this email is not spam. He creates a transaction with his MailCoin wallet, giving back the MailCoins to Alice. He gives the MailCoins back because he benefits more from giving them back than from holding onto them. If he would hold onto them, Alice would not want to communicate with him in the future and Bob could miss out on important information [15].

5.1.2 Communication between dishonest communication partners

In this scenario, the spammer Eve wants to send Bob an email. Eve has to attach a valid email stamp to the email because otherwise the email would not even reach Bob's inbox. Eve writes the email, attaches a valid email stamp, and sends the email to Bob. Bob receives the email because it has a valid email stamp attached and reads it. Bob now needs to decide whether he thinks the email is spam. If he does not think it is spam, he can send back the MailCoins, as he did with Alice. If he thinks, the email is spam he keeps the funds and deletes the email. Because Eve actually did send spam, Bob decides the email is spam. Eve loses the funds she attached to the email. Sending unsolicited emails would only make sense, if it was worth enough

for the sender. This would, for example, apply to spear phishing attacks.

5.1.3 Mass email services

Traditional proof-of-work systems do not work with mass email services like newsletters or email lists, as these services send many emails in a short time span, which is also the behavior of spammers [13]. A solution to this problem would be to add these services to a white list [15]. In the MailCoin anti-spam system, these services are still possible, even without a white list. To subscribe to the service the subscriber pays a MailCoin to the service. The service uses this MailCoin to send emails to this subscriber. To stay subscribed, the subscriber is required to send back the MailCoin after each received email. This scenario does not yet include transaction fees. Normally, the sender of the emails would have to pay transaction fees, but because the subscriber is the one actually profiting from using the MailCoin anti-spam system, the subscriber should have to pay. Therefore, the subscriber not only sends the received MailCoins back but also the expected transaction costs.

5.2 Implementation

The aim of the prototypical implementation is to make the system as transparent as possible for the users and to integrate easily into existing systems. It consists of two parts: an add-on for the email client Thunderbird and a plug-in for the email spam filter SpamAssassin. The implementation furthermore consists of a connector between these extensions and the MailCoin wallet, and the MailCoin blockchain. This connecting program is called "MailCoin connector". It acts as an abstraction layer for MailCoin wallet and blockchain functionality needed in the end user programs. This modular approach allows the replacement of components without breaking the system. It is possible to replace the used blockchain solution without changing anything in the other components. To use Bitcoin, for example, instead of MailCoin, the only change that needs to be made is to start the Bitcoin node instead of the MailCoin node and reconnect the MailCoin connector. The modular approach also allows for easier development, as the functionality is bundled in one program and not spread over different programs. The connector connects to the MailCoin blockchain and wallet via the MailCoin RPC and offers a connection over WebSockets, which allows the other components to connect. It is programmed in Go and can use libraries from the MailCoin client, which is also programmed in Go. The extensions must follow their parent program specification; the Thunderbird add-on is written in JavaScript and the Spamassassin plug-in in Perl.

5.2.1 Thunderbird add-on

The Thunderbird add-on integrates MailCoin anti-spam functionality into Thunderbird. It serves two purposes: users can create and attach email stamps to their messages and check the validity of the email stamps of received emails.

The functionality at the sending side of the add-on adds a button to the message compose window of Thunderbird. When clicked, it opens the windows shown in Figure 5.2. When the sender enters the P2PKH address of the receiver and clicks the "Create transaction" button the add-on creates the "MailCoin-Header" (see Chapter 3) for the email and attaches it to the email. It then sends the address and the hash of the "MailCoin-Header" with the command to create a transaction to the connector. The connector creates the transaction with a MailCoin output and a P2PKH output to the receiver. It then sends the ID of the created transaction back to the add-on, which attaches the transaction ID to the email. After this process, the email contains two additional email headers, namely the "MailCoin-Header" and the "MailCoin-TxID" header, and the transaction is published to the network. The sender then sends the email like any other email.

On the receiving side, the add-on adds a "Check MailCoin transaction" button to the email display window. When this button is clicked, the add-on searches the email for MailCoin headers. If there are MailCoin headers the plug-in checks whether the data in the "MailCoin-Header" corresponds to the data in the email. It checks whether the time stamp and the receive time are similar and whether the receive address is the same in the "MailCoin-Header" and in the email. If all these checks are completed successfully, the add-on sends a hash of the "MailCoin-Header" and the "MailCoin-TxID" with the command to verify it, to the connector. The connector verifies the email stamp and sends back a status code, which is either a success or an error code. Depending on the return code, a message box is displayed to the user. This feature should only be used for a second check. Normally, a spam filter would do the filtering of messages and the user would only receive messages that are already successfully checked. The spam filter is described in the following section.

5.2.2 Spamassassin plug-in

The SpamAssassin plug-in integrates MailCoin functionality into the email spam filter SpamAsssassin. It allows SpamAssassin to check the validity of email stamps by querying the MailCoin connector, which queries the MailCoin blockchain.

The plug-in adds a new filtering rule to SpamAssassin. Every email is checked for MailCoin headers. The information in the MailCoin headers is collected and sent to the connector to verify. The procedure is the same as

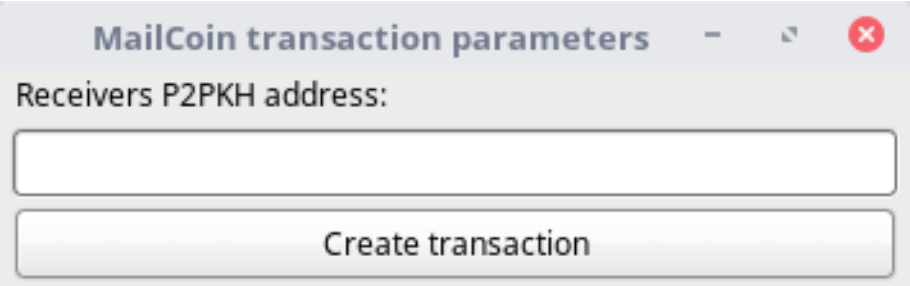


Figure 5.2: This window pops up, when the "MailCoin" button is clicked in the Thunderbird compose window. It makes it possible to create an email stamp and to attach it to the email.

in the Thunderbird plug-in, at least on the receiver's side. The plug-in can only exit with either "success" or "failure". "Success" means the email stamp is valid, and "failure" means that either there is no email stamp or it is not valid. Users have to configure their spam filters. They could either configure that emails that pass the check are less likely to be spam and lower the spam score, or that emails that do not pass the check are more likely to be spam and raise the spam score. In the first scenario, it would be easier for emails to not be classified as spam, if they have an email stamp attached. Emails not containing an email stamp would not be affected by this plug-in. In the second scenario, it would be harder for emails which do not contain an email stamp to pass the spam filter without being classified as spam. The users can also define the change in the spam score, depending on the outcome and the required spam score to mark an email as spam.

5.2.3 MailCoin connector

The MailCoin connector is a Go program which contains most of the functionality needed by the plug-ins. When started, it tries to connect to the MailCoin wallet. It queries the current balance and creates a new MailCoin address for receiving MailCoins. The current balance and the new address are printed out. Other programs can connect to it on port 1337, over the WebSocket protocol[1]. It accepts various commands on this port that trigger the execution of various functions.

To create an email stamp, programs need to issue a "CreateMailCoin-Transaction" command. This command needs two parameters: the SHA-256 hash of the "MailCoin-Header" and the P2PKH address of the receiver. The function creates a MailCoin transaction containing a MailCoin output with the SHA-256 hash of the header and an output sending MailCoins to the address specified. This transaction is funded automatically with an-

[1] https://tools.ietf.org/html/rfc6455

other transaction from the wallet. It is then signed and published and the transaction ID is sent back to the client requesting this operation.

There are two functions that verify an email stamp. One verifies email stamps by querying the wallet, and the other verifies email stamps by querying the blockchain. The wallet can only be queried for transactions concerning the wallet. This is enough for end users who receive emails and transactions. The used implementation does not support watch-only addresses. Due to this, the spam filter has to query the blockchain directly. This requires the MailCoin daemon to maintain a full transaction index. The function "CheckMailCoinTransaction" queries the wallet and the function "CheckRawMailCoinTransaction" queries the blockchain directly. Calling "CheckMailCoinTransaction" and "CheckRawMailCoinTransaction" works the same. Both need two parameters, the hash of the "MailCoin-Header" and the MailCoin transaction ID. "CheckMailCoinTransaction" gets the raw transaction from the wallet, searches for the MailCoin output and checks the equality of the hash. "CheckRawMailCoinTransaction" does the same, however, it gets the raw transaction from the MailCoin daemon, which holds the information for all transactions in the blockchain.

5.3 Requirements for this use case

5.3.1 Technical requirements

The technical requirements are the basic requirements of this use case. Later on, the requirements for this use case are discussed, in order for the concept to make sense. For this use case to work, both the sender and the receiver have to use MailCoin or another blockchain based cryptocurrency that at least supports basic scripts. Besides the receiver's email address, the sender has to know the receivers P2PKH address and how many MailCoins they at least have to attach so the receiver considers an email stamp as valid. The sender needs MailCoins to pay the transaction fees and to pay for the P2PKH output sending MailCoins to the receiver. To accomplish this, the sender needs a wallet software which can create and publish transactions.

The receiver needs a wallet that monitors the MailCoin blockchain for incoming transactions. If the receiver wants to filter emails with an external spam filter, the spam filter needs to download the whole blockchain as well, so it can search all transactions. If the spam filter is trusted, the receiver could simply use blockchain data from the spam filter and would not have to synchronize the blockchain on their computer.

5.3.2 Blockchain requirements

The system can theoretically be based on any blockchain, but some blockchains are better suited for the requirements of the system. In the basic

system, every email should have an attached transaction. This means transaction fees are email fees. These fees should be as low as possible, because fees, in contrast to the transaction value, cannot be refunded. A blockchain with high fees is not a practical basis for the MailCoin anti-spam system.

It is not safe to accept transactions without confirmations. This also means that only email stamps that have at least one confirmation should be considered valid. It is necessary to wait for this confirmation, either on the sender or on the receiver side. Blockchains with a long time between blocks would slow down the communication. This is not as problematic as the transaction fee requirement, but high confirmation times are still not optimal for the system. To lessen the impact, the sender could already publish the transaction when starting to write the email and send it after the transaction is confirmed.

The MailCoin cryptocurrency tries to solve both these problems by reducing the time between blocks but keeping the maximum block size. This allows a higher throughput, because now blocks are generated more often, which should keep the transaction fees low, as there is less competition to get miners to include a transaction in a block. Furthermore, a lower mean time between blocks allows for faster transactions confirmations.

5.3.3 Requirements on the users

When a receiver gets an email with an attached transaction, there is no mechanism preventing them from keeping the transaction value. It would be beneficial for the system if users sent back the value of the transaction, because that way the transaction would not be a payment but a bond. Normally, users send back the funds because it is in their own interest. Keeping the funds would hurt their trustworthiness and other users would stop communicating with them [15]. Even if this was not the case, only users who send more emails than they receive would eventually have to pay. In a traditional proof-of-work system, it is hard to define how much work is required; it has to be high enough to be unprofitable for spammers, yet at the same time low enough to still be usable for honest users [13]. If the transactions referenced in email stamps were used as bonds, this problem would be solved completely. The bond could be high, because it would be paid back anyway. In this scenario, only spammers had to pay. If they were not used like bonds, users would keep the funds and the problem would still be solved partially, because honest users would be paid back, when they get an answer or another email with an attached transaction. The spammer's money is lost in both cases.

5.3.4 Other requirements

In the scenario that only a few users use the system, they can only communicate in this small group. In a very small group, they could just add each other to a white list and not use the system at all. A higher number of users is required for it to make sense to use the system, because it would not be viable anymore to just have a list of users to trust.

For the system to work perfectly, all honest email users would have to use this system. Every email would have an attached transaction and only emails with an attached transaction would be accepted. This would not completely solve the problem of spam, but at least drastically increase the costs of sending spam. Because everybody would use the system, communication without the system would not be possible anymore.

This shows that a very high percentage of users is required. Even with a configuration that allows communication with people who do not use the system, the communication with non-users is more complicated.

Another requirement is a way to communicate MailCoin addresses and minimum MailCoin costs. Email addresses normally do not change that often, MailCoin addresses should in fact be changed after every transaction. Because of the expected volatility of the exchange rate, the minimum MailCoin costs should also be easily adaptable. This can be done in different ways, for example also face-to-face, but for usability purposes, it would be good to have a system that simplifies this. How such a system could be introduced is described in Chapter 8.

5.4 Advantages for users

The biggest advantage for users is that they will receive a lot less spam. For the spam they still receive, they will be paid for.

The private and public keys, which are generated when MailCoin addresses are generated, can be used to encrypt and sign emails. Signing is always possible with the private key that also signs the attached transaction. This only proves the integrity of the email, because the receiver never received the public key from the sender in person, but only from the transaction referenced in the email stamp. The actual sender could be anybody who has access to the email account. If subsequent emails were signed with the same private key, this would prove that the emails all come from the same sender or at least somebody who also has control over the MailCoins. It is not possible to encrypt the first message to another user, because in this situation only the MailCoin address of the receiver is known, but not their public key. As soon as a message with an email stamp is received, every subsequent email can be encrypted with the public key published in this transaction. This functionality can be used with the normal MailCoin system without adding any complexity.

5.5 Disadvantages for users

The system adds complexity for the users. They cannot just send emails, they also have to create the MailCoin transactions, keep the blockchain synchronized and run the MailCoin programs. To create the transactions it is also necessary to know the MailCoin address and the amount of MailCoins to send. Distribution of this information adds complexity for the users as well.

Furthermore, using the system adds costs for the users. Running the system costs money because it needs a lot of resources. Especially this use case causes a lot of transactions in the blockchain, which cause a lot of blockchain data that users have to download and store.

Users also have to pay the transaction costs. The system raises the cost for emails and users will only pay that price as long as the advantages of using the system are higher than the price paid for using the system. Furthermore, users lose the transaction value if the receiver does not pay them back, which could happen occasionally.

Another disadvantage for the users is that their privacy is weakened when they use the system. Transactions on the blockchain are public. Everybody who can see the emails, for example, an intelligence agency, can also check the MailCoin transactions. This third party then knows which MailCoin addresses belong to which user. The third party can then track the funds used in this transaction and can see which users communicate with each other. This can be prevented by using a new address for every transaction or at least every communication partner. The information in the blockchain is saved publicly. Even though the transactions are between pseudonymous identities, these identities could be assigned to the real person at any later point, as the transaction data is saved forever.

Communication between users of the system becomes easier than usually, communication between users and non-users becomes harder, because users who do not use the system need to be on a white list or the system has to be configured to still accept emails without email stamps. Accepting emails without emails stamps would weaken the system. So, a disadvantage is that communication with non-users is still possible but much harder than normally.

If the system works as expected, users would not be used to spam anymore and would be more trusting. This could actually make spam, if it has managed to reach the users, more effective.

5.6 Problems

First, users need to ask themselves whether it is even worth it to use the system. They can primarily solve the problem of spam by using it. To run the

system, they need a lot more resources and they have to pay the transaction fees. This alone seems like a bad exchange.

As described earlier, the system only really works if everybody uses it. The adoption rate of a new system normally rises gradually and it attracts more users over time if people like it. MailCoin is not really usable with just a few users. It would most probably not be possible to instantly get a high adoption rate.

One of the benefits of this system are low transaction fees. This is certainly achievable if the transaction volume is low. Another requirement is a large number of users. A large number of users means many transactions. This could be achievable with a higher maximum block size, but it would lead to a bigger blockchain size and is not guaranteed to solve the problem. It would be very hard to keep the transaction fees low while having a high transaction volume.

5.7 Adapted use case

Because of the problems with the presented use case, it is not viable with this technology. Some of the problems could be fixed by implementing new features for the technology. They could make the use case interesting again. (see Chapter 8)

This section describes possible changes that create new, similar use cases that have fewer problems and are more viable. A minor change would be to use the presented system like a payment system. Buyers would write emails detailing what they want to buy, and attach a MailCoin transaction with a P2PKH output of the item value. This would be an easy system for buying and selling products online. Usually, the seller needs a secure payment system on their website. With this system, the seller's website could be static, just showing products, their prices, a contact email address, and a payment address. For small online businesses this would be great, because they would not need a lot of IT infrastructure. Especially for digital goods, this would be an interesting system because the seller could automatically process incoming emails and the attached transactions and then automatically send the products. In this system, all the problems described above would not apply, because in this scenario MailCoin would only be used to pay for real life products.

Another use case based on the anti-spam solution would be something called premium-rate emails. These premium-rate emails work like premium-rate phone numbers. Senders have to pay so their email will be received. The receivers of the emails automatically delete all emails that do not have an email stamp attached with at least the minimum amount of payment. This works like in the original use case. The difference is that senders send a payment and do not expect to get their money back. This payment is for

using the service. The receiver only reads and potentially answers emails that have a payment attached. This change has a big effect on the requirements and the problems. Now only the receiver has to constantly run the whole system, the senders just need to run it for one transaction. There is no need for a high number of users anymore. The transaction fees are now seen as part of the payment and are only needed for one email; this means it would not be as severe if they were high. Transaction confirmation times are also not as important, because the system is not used to send messages back and forth. Because of the lower requirements, this use case is possible with a high number of cryptocurrencies, which would make it much easier to get started.

Chapter 6

Receive confirmations

The MailCoin system for linking emails with transactions in a blockchain can be used to implement a system for receive confirmations for emails. One important feature of these receive confirmations is that they are not optional. A receiver who does not confirm reception of an email, cannot read the email. This system comes with the same benefits for the communication partners as the service "Einschreiben" by the Austrian "Post AG". [1] A sender who pays for this service will receive a receive confirmation once the receiver accepts the mail. The receiver has to confirm receiving the mail by signing a document. If the receiver does not sign, they do not receive the mail and the sender will get the information that the mail was rejected. The advantages for the sender are that they can later prove whether the receiver accepted or rejected the mail. An advantage for the receiver is that they can later prove that they received or did not receive the mail. In the traditional postal system, this only works because there is a trusted third party who is bound by the "Postgesetz" [4]. With MailCoin, this system is possible without a trusted third party.

6.1 Receive confirmations with MailCoin

In this scenario, Alice wants to send Bob an email and she wants a receive confirmation for this email. First, Alice needs Bob's email address, a public key, and a MailCoin address of Bob. The public key and the address do not have to belong together, but if they do, Alice only needs Bob's public key, because she can calculate the address using the key. Alice then writes the message m she wants to send to Bob and she also chooses a random 256-bit key k. Alice sends some MailCoins to a 2-of-2 multi-signature address. This transaction can only be spent if it has two signatures, one with Alice's private key and one with Bob's private key. Alice creates a transaction with

[1] https://www.post.at/downloads/PB_Brief-Zusatzleistungen_final.pdf

this multi-signature transaction as input and an output which sends funds to Bob's address. It also contains a MailCoin output with a hash of the encrypted message *m*. Alice signs this transaction. Because it uses the multi-signature transaction as input, it needs a second signature from Bob. This transaction is called *T1*. Alice creates the email for Bob. This email contains the raw transaction *T1* and the message she writes to Bob, encrypted with *k*. Alice also encrypts this email with Bob's public key. She then optionally adds an email stamp to this email and sends the email.

Bob receives the email and decrypts it with his private key. Bob has the half-signed multi-signature transaction *T1*, the encrypted message *m* and a hash of encrypted *m*. Bob cannot read the message, because he does not know the encryption key *k*. If Bob did not continue, it would be the same as refusing to sign the receive confirmation in the traditional postal system. Bob could not get any information about the message out of that email. To further participate in the protocol, Bob signs *T1* with his public key. *T1* contains two signatures now and is valid. Bob publishes *T1*, signaling that he received the email and because of Alice's signature this proves that both parties participated. This is equivalent to signing the confirmation of receiving in the traditional postal system.

As soon as Alice sees *T1* confirmed in the blockchain, she creates a transaction which contains *k*. This is a transaction with a MailCoin output, but, instead of the SHA-256 hash of the MailCoin-Header, it contains *k*. It also contains an output sending MailCoins to Bob. This transaction is called *T2*. Alice publishes *T2*, publishing the key *k*. This is the equivalent of handing the letter over. If Alice stops the protocol, Bob can prove that he received the email, but Alice did not give him the encryption key for the message. He can prove that because there is a published *T1*, but no *T2*. Bob can not only prove that, he also receives some MailCoins for his involvement, because *T2* sends MailCoins to Bob. This is a compensation for Bob.

Bob must wait for *T2* to be published in the blockchain. Because *T2* contains an output giving MailCoins to Bob, he has to scan for incoming transactions. He does not have to wait for the transaction to confirm, because he only needs the *k*, which is included in the transaction. Bob extracts *k* from the transaction. *k* is now publicly known, but only Alice and Bob have the cipher text. Bob uses *k* to decrypt the cipher text of *m*. This is the equivalent of opening a letter. Bob now has the message *m* and can read it.

Both communication partners can prove that they participated in the protocol by showing the public transactions on the blockchain. Alice could publish an incorrect *k* in *T2*. In this case, Bob can just publish the encrypted *m* and everybody can verify the incorrectness by trying to decrypt it, with the publicly available *k*.

This protocol is also shown in Figure 6.1.

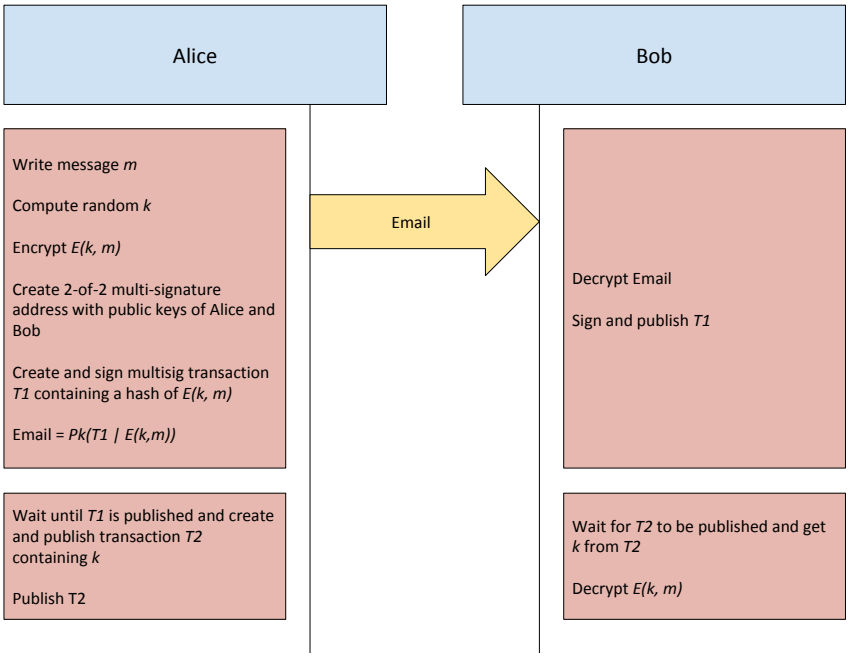


Figure 6.1: The MailCoin receive proof system.

6.2 Requirements for this use case

The basic requirement for this use case is a cryptocurrency that allows basic scripts. P2PKH transactions, P2SH transactions and transactions with data only must be possible.

Low transaction fees would be an advantage, but not required. The service "Einschreiben" by the Austrian "Post AG" costs 2.20 €. The protocol requires at least two transactions and one transaction for the email stamp. To give an incentive to the receiver to participate, the transactions should also contain a payment. The sum of these costs should be lower than 2.20 € to be able to compete with the traditional postal system. These fees are paid by the sender, like in the traditional postal system.

Confirmation times are not that important, but low confirmation times would be an advantage. The entire system is slowed down by high confirmation times, but the confirmation times just impact the speed of emails with receive confirmations. The requirement here is similar to the basic MailCoin requirement.

The sender needs to use the system. The receiver is not required to use the system initially, because the email can be received without using MailCoin. To decrypt the unencrypted message and get the information, the receiver has to participate in the MailCoin system. The receiver not

only has the incentive to participate to get the message, but also to get the attached MailCoins. Because of these characteristics of the system, there is no minimum number of users required.

6.3 Advantages for users

The advantage for the sender is that the system provides mandatory receive confirmations without a trusted third party. Normally, either a third party has to be trusted or the receive confirmation is optional, allowing the receiver to read the message without sending a receive confirmation. This is not possible here.

The receiver has to participate in the protocol to receive the message. An incentive to do so are MailCoins attached to the protocol transactions that the receiver can only get by participating in the protocol. After completing the protocol, they can also prove that they received exactly this message. The sender has to pay to use the system, but the receiver can get money by using the system.

6.4 Disadvantages for users

Running the system causes overhead. In this use case, it is minimized because initially, only the sender has to run the system. The receiver has to run the system to participate. There is no need to have the system always running in the background, like the anti-spam system. Because not all emails need an attached transaction, there are fewer transactions in the blockchain, effectively keeping the total blockchain size low. Fewer transactions would also keep the transaction fee low.

The system is not free to use, at least not for the sender. The service "Einschreiben" by the Austrian "Post AG" can be seen as a direct competitor, because it offers the same advantages. As long as the use of this system is cheaper than the use of the service "Einschreiben", it would be the better choice for senders. The MailCoin system for receive confirmation also has the advantages that, while the protocol is not instant, it is usually faster. The protocol can be completed in the time it takes to generate a block, because only the first transaction *T1* has to be confirmed. There is also no third party and no requirement of trust involved, which can be an advantage.

6.5 Conclusion

The MailCoin system for receive confirmations offers users a system similar to "Einschreiben" in the traditional postal system. The system offers a wide range of advantages, while not causing many disadvantages for the users.

There are not a lot of requirements, so the system could be used with most major cryptocurrencies. The fact that there are not major problems and that the advantages outweigh the disadvantages result in the conclusion that the system is practicable.

Chapter 7

Requirements for a successful cryptocurrency

The cryptocurrency MailCoin is a part of this project. It was developed to meet the requirements of the use cases and would be a good basis. The anti-spam use case especially has strong requirements and would not work well with most other cryptocurrencies. This chapter explains why other cryptocurrencies are successful, what could be done for MailCoin to be successful, and what the prospects of success are for MailCoin. This thesis defines a successful cryptocurrency as a cryptocurrency with a high market capitalization (market cap)[1].

7.1 Successful cryptocurrencies

Narayanan et al. describe in [19] how launching a cryptocurrency works. There are several types of stakeholders in a cryptocurrency and for a cryptocurrency to be successful, it needs to attract all of them. According to [19] the stakeholders in Bitcoin are developers, miners, investors, merchants, customers, and payment services. All these stakeholders are important and interrelated and there is no simple recipe to attract them. These stakeholders have to be convinced that the cryptocurrency is worth investing time and money. A cryptocurrency needs a good narrative describing the innovative features it brings, the new possibilities it opens, and why people should invest in it [19].

coinmarketcap.com lists cryptocurrencies by market capitalization. This section examines the ten cryptocurrencies with the highest market capitalization and their unique selling proposition, trying to get an idea who the

[1] The market capitalization of a cryptocurrency is the value of the whole market of that cryptocurrency at a certain time. It is calculated by multiplying the number of existing coins with the current price per coin.

competitors are and what a successful cryptocurrency needs.

The ten cryptocurrencies by market capitalization on coinmarketcap.com are: (retrieved on May 3, 2017)

1. Bitcoin[23]
2. Ethereum[4]
3. Ripple[5]
4. Litecoin[6]
5. Dash[7]
6. Ethereum Classic[8]
7. NEM[9]
8. Monero[10]
9. Golem[11]
10. Augur[12]

7.1.1 Bitcoin

Bitcoin was the first cryptocurrency and it is advertised as a decentralized, fast, peer-to-peer payment network with low fees [3]. Chapter 2 describes the basics of Bitcoin in detail. Because Bitcoin was the first cryptocurrency, it did bring lots of new functionality at the time.

7.1.2 Ethereum

Ethereum is advertised as a decentralized platform that runs smart contracts. It has a more extensive scripting language than Bitcoin, which allows more sophisticated smart contracts [10].

7.1.3 Ripple

Ripple is a global settlement network which aims to change the way in which money is sent. Ripple primarily seeks to reduce transaction fees between banks. It provides cheap, fast and traceable transactions [26].

[2] https://bitcoin.org/en/
[3] https://www.bitcoin.com/
[4] https://www.ethereum.org/
[5] https://ripple.com/
[6] https://litecoin.com/
[7] https://www.dash.org/
[8] https://ethereumclassic.github.io/
[9] https://www.nem.io/
[10] https://getmonero.org/home
[11] https://golem.network/
[12] https://augur.net/

7.1.4 Litecoin

Litecoin is very similar to Bitcoin. The main differences are a shorter time between blocks and a different mining algorithm [14]. Because Litecoin is the first, still active alternative cryptocurrency, there was no competition and it did not have to bring a lot of new features.

7.1.5 Dash

The cryptocurrency Dash is advertised as a cryptocurrency that allows private and instant payments. Dash promises more privacy by using a self-developed protocol called "PrivateSend". This is an improved and extended version of "CoinJoin". The basic principle is that users group their transactions together into one transaction. A third party does not know which outputs belong to which inputs [7]. For instant transactions, Dash implements a system called "InstantSend". It allows for faster transaction by reaching consensus on these "InstantSend" transactions before they are included in a block [8].

7.1.6 Ethereum Classic

Ethereum Classic is a continuation of the original Ethereum project. The original Ethereum project did a transaction rollback. This created a fork: Ethereum Classic retains these rolled back transactions and Ethereum discarded these transactions. It was expected that the old chain would quickly die off, but miners kept mining and so two blockchains were created. Besides the different blockchain from block 1920001, Ethereum Classic did not implement any big changes. Until block 1920001, Ethereum Classic and Ethereum have the same blockchain. This means that everybody who owned Ether before this block afterwards owned them in both blockchains [9].

7.1.7 NEM

NEM introduces new blockchain concepts. It uses the proof-of-importance consensus algorithm, which allows users with many transactions to generate blocks. In contrast to proof-of-work, this algorithm does not waste energy. NEM also uses a reputation system called "Eigentrust++" and other improvements [20].

7.1.8 Monero

Monero is a cryptocurrency that focuses primarily on privacy. Similar to Dash, it offers private transactions. One major difference is that Monero is not based on Bitcoin but on the CryptoNote protocol [27].

7.1.9 Golem

Golem is built on the Ethereum platform; thus, its value is also related to Ethereum. It creates a decentralized peer-to-peer market for computing power. Users can rent out and rent computing power with Golem [24].

7.1.10 Augur

Augur is a prediction market. Like Golem, Augur is built on the Ethereum platform and its value is also related to Ethereum. The Augur prediction market allows users to bet on the outcome of events and earn money by trading those predictions. It also aims to predict events with this collective wisdom [21].

7.1.11 Summary

All of these successful cryptocurrencies brought something completely new, an improved way of doing something, or have another unique sales characteristic. Especially those that are primarily a cryptocurrency and not a network (Bitcoin, Ethereum, Litecoin, Dash, Ethereum Classic and Monero) had been around a long time before their value and market capitalization rose. This is because they needed to acquire user trust by not having major security problems and being used.

7.2 Success of MailCoin

MailCoin is very similar to Litecoin. Both lower the mean time between blocks; Litecoin additionally changed the mining algorithm. The real uniqueness of Litecoin is that it was the first alternative cryptocurrency. Having this in mind, the MailCoin cryptocurrency cannot bring many advantages in 2017.

The MailCoin system for linking emails was designed to work with every cryptocurrency; it does not need the MailCoin cryptocurrency. The two use cases presented in this thesis work well with the MailCoin cryptocurrency. Specifically, the MailCoin anti-spam system (see Chapter 5) requires a cryptocurrency with fast confirmation times. Not a lot of cryptocurrencies provide that. MailCoin seems like the obvious choice here. As also described in Chapter 5 there are still a lot of problems with the MailCoin anti-spam system, which damages the usability of the use case. Currently, this use case does not seem to be viable. The adapted use case, also described in Chapter 5 seems to be viable, but it does not necessarily require the MailCoin cryptocurrency. The MailCoin receive confirmations (see Chapter 6) also do not necessarily require the MailCoin cryptocurrency.

To make MailCoin a successful cryptocurrency, it would need a lot more

development time, more advertising and use cases. This could be accomplished but is outside of the scope of this thesis. For now, the MailCoin cryptocurrency is a concept to demonstrate that the use cases described here can be implemented.

Chapter 8

Future work

This chapter is about features that were not implemented in this thesis, because they are outside its scope. It is about possible changes to the existing system, new possible features for the system, and completely new systems that could improve the use cases.

8.1 "Segregated Witness" and the "Lightning Network"

Segregated Witness is a change in the protocol of Bitcoin based cryptocurrencies. The implementation of "Segregated Witness" brings a range of new features. For MailCoin especially the fix for transaction malleability is interesting. Transaction malleability is the problem that the transaction hash of a signed transaction could be changed before it is confirmed in the blockchain without invalidating the signature. Because of transaction malleability, it is not safe to use unconfirmed transactions as the input for other transactions [2]. This, in turn, is necessary for the lightning network. The lightning network is a system for payment channels. Once a payment channel between two parties is opened, it allows transactions between those two parties without publishing the transactions on the blockchain. This would allow instant transactions without transaction fees. Only to open and close the channel a transaction would be required [22]. The anti-spam use case (see Chapter 5) especially needs fast transactions with low transaction fees. The lightning network could solve these problems for the anti-spam use case.

8.2 System for addresses

Currently, it is not defined how the exchange of MailCoin addresses works. The communication partners could exchange them when they exchange email addresses, but normally email addresses stay the same over a long time. For the system to work, it would be fine to always use the same MailCoin address. However, this is a privacy problem. A new MailCoin address

should be generated for every transaction. If Bob always uses the same address for all transactions he receives, Alice, who knows Bob's address, can see all the transaction Bob receives. This becomes especially problematic when Alice sends coins to Charlie and later sees that these coins are sent to Bob's address. She now knows that Charlie sent these coins to Bob.

Senders would have to ask for a new MailCoin address every time they wanted to send a new email. For the anti-spam system, not only the addresses change frequently, but also the number of MailCoins that needs to be sent with an email stamp to be considered valid. Exchanging this information manually would be impractical. A system is needed to assist the users in retrieving and updating this information. The system could be similar to DNS, where clients can query the MailCoin address and minimum MailCoin amount belonging to an email address. Users who trust their own email servers could run this service with their private keys on their email servers. The service could then generate a new private key for every request.

8.3 MailCoin cryptocurrency

As concluded in Chapter 7 the MailCoin cryptocurrency could most likely not be successful right now. It needs further development. As described earlier in this chapter, "Segregated Witness" could be implemented. MailCoin also needs more unique selling propositions. It needs more features and a more user-friendly implementation.

8.4 Implementation of use cases

Currently, the described use cases are implemented as prototypes. The implementation is not usable by end users. For the use cases to get adopted, the usability of the implementations has to be improved and features must be added. The not yet implemented use cases, which only exist as concepts should be implemented as well.

8.5 Further use cases

The use cases described in this thesis are examples. Based on the system for linking emails with transactions, furthermore use cases would be possible. The development of more use cases would be crucial for the MailCoin cryptocurrency to succeed.

8.6 Third party analysis

Even though this thesis already includes lots of received feedback, the developed systems were never fully analyzed by a third party. The described system should be seen as a first version; this version should be verified and improved if possible.

Chapter 9

Conclusion

9.1 Answering the research question

The research question is:

> How is it possible to uniquely link emails with transactions of a cryptocurrency and how can an anti-spam system and a system for receive confirmations be implemented on top of that?

The research question is answered by answering the sub-questions.

> 1. Sub-question: How can emails and cryptocurrency transactions be uniquely linked together?

Chapter 3 describes the system for uniquely linking email with cryptocurrency transactions. To link emails to cryptocurrency transactions, the system adds two new headers to emails, the "MailCoin-Header" and the "MailCoin-TxID". The "MailCoin-Header" contains information about the email, such as the recipient and a time stamp. The "MailCoin-TxID" is the transaction ID of the linked cryptocurrency transaction. For the link to work both ways, this transaction contains a hash of the "MailCoin-Header" of the email.

> 2. Sub-question: What are the requirements for a blockchain used in this system?

Section 3.3 talks about these basic requirements, which all also apply to all of the use cases. The system is designed to work with most cryptocurrencies, but it works best with cryptocurrencies with a low mean time between blocks and a low transaction fee.

> 3. How can the anti-spam system and the system for email receive confirmations be implemented?

The anti-spam system is described in Chapter 5 and the system for receive confirmations is described in Chapter 6. Both systems rely on the linking of emails and cryptocurrency transactions.

> 4. Sub-question: What are the requirements for these use cases to work?

Both use cases have the same basic requirements as the system for linking emails and transactions. The anti-spam system especially needs low transaction fees and a low mean time between blocks. The requirements are described in detail in Sections 5.3 and Section 6.2.

> 5. Sub-question: What are the advantages for users of the systems?

Users of the anti-spam system receive less spam, and for the spam they still receive they get paid. The system for receive confirmations allows users to prove the reception of emails. This is described in detail in Chapter 5 and Chapter 6.

> 6. Sub-question: What are the disadvantages for users as opposed to not using these systems?

Using the system generates public, non-erasable transactions. These could be used to monitor the communication of users. Using the system also generates costs: cost for transactions and costs for operating the system. The disadvantages for the specific use cases are described in detail in Section 5.5 and Section 6.4

> 7. Sub-question: Are there other use cases that can be based on MailCoin?

MailCoin can also be used as the basis for a system to buy products by email in which the payment for the product is directly attached to the email. Another use case is premium-rate emails. These use cases are described in Section 5.7.

> 8. Sub-question: How can the MailCoin cryptocurrency be successful?

The MailCoin cryptocurrency needs to provide the users with something new. Currently, it is not able to compete with other cryptocurrencies, because MailCoin does not offer enough new possibilities. To be a successful cryptocurrency, MailCoin would need much more development: implementing new features and developing more use cases, as well as, more advertisement to inform users of its existence. This is described in detail in Chapter 7.

9.2 Outlook

The usefulness of the system largely depends on its features. The anti-spam use case described in Chapter 5 does not seem usable in this form because there are many problems. The adapted anti-spam use case, premium-rate emails, is a system with lower requirements. Because of the lower requirements, it does not have the problems the anti-spam system has. The same functionality is already used for phone calls; it seems natural that this functionality would also be used for emails. The system for receive confirmations (see Chapter 6) offers the same advantages as the service "Einschreiben" by the Austrian "Post AG". Because this service is used for the traditional postal system, it would only be logical that it would also be used for emails. Both these use cases provide the users with something new to the users that is already possible and popular in other fields.

The MailCoin cryptocurrency has not been developed sufficiently at this time. It would not make sense to release it publicly now. The promising use cases do not have strong requirements and could be deployed using most other cryptocurrencies. It would make sense to implement them using an already existing and established cryptocurrency.

Before the systems described here are used in a production environment, they should be reviewed by a third party to prevent potential loss of funds or other problems.

Appendix A

CD-ROM/DVD Contents

A.1 PDF-Files

Pfad: /

mailcoin.pdf	Master thesis

A.2 Source code

Pfad: /source_code

mlcd.zip	MailCoin daemon
mlcwallet.zip	MailCoin wallet
MailCoinConnector.zip .	MailCoin connector
ThunderbirdAddon_MailCoin.zip	Thunderbird add-on
SpamassassinPlugin_MailCoin.zip	Spamassassin plug-in

A.3 Literature

Pfad: /literature

1.pdf	Martín Abadi et al. “Bankable Postage for Network Services”
2.pdf	Bitcoin Core :: Segregated Witness Benefits
3.pdf	Bitcoin — Open source P2P money
4.pdf	“Bundesgesetzblatt 2009/123”
5.pdf	Vitalik Buterin. Toward a 12-second Block Time
6.zip	Developer Documentation - Bitcoin
7.pdf	Evan Duffield and Daniel Diaz. “Dash: A Privacy Centric Crypto Currency”

8.pdf . . . . . Evan Duffield, Holger Schinzel, and Fernando Gutierrez. Transaction Locking and Masternode Consensus: A echanism for Mitigating Double Spending Attacks

9.pdf . . . . . Ethereum Classic

10.pdf . . . . . Ethereum Project

11.pdf . . . . . Darya Gudkova et al. “SPAM AND PHISHING IN Q2 2016”

12.pdf . . . . . Tony Hansen and Gregory M. Vaudreuil. Message Disposition Notification

13.pdf . . . . . Ben Laurie and Richard Clayton. “Proof-of-Work” proves not to work; version 0.2.

14.pdf . . . . . Charlie Lee. [ANN] Litecoin - a lite version of Bitcoin. Launched!

15.pdf . . . . . Thede Loder et al. “The Spam and Attention Bond Mechanism FAQ”

16.pdf . . . . . Microsoft. The Penny Black Project

17.pdf . . . . . Keith Moore. SMTP Service Extension for Delivery Status Notifications

18.pdf . . . . . Satoshi Nakamoto. Bitcoin 0.3.2 released

20.pdf . . . . . NEM - Distributed Ledger Technology (Blockchain)

21.pdf . . . . . Jack Peterson and Joseph Krug. “Augur: a Decentralized, Open-Source Platform for Prediction Markets”

22.pdf . . . . . Joseph Poon and Thaddeus Dryja. “The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments”

23.pdf . . . . . Justin M Rao and David H Reiley. “The Economics of Spam”

24.pdf . . . . . The Golem Project

25.svg . . . . . Theymos. File:Blockchain.svg - Wikimedia Commons

26.pdf . . . . . Welcome to Ripple | Ripple

27.pdf . . . . . [XMR] Monero - A secure, private, untraceable cryptocurrency

References

Literature

[1] Martín Abadi et al. "Bankable Postage for Network Services" (2003), p. 20 (cit. on p. 4).

[2] *Bitcoin Core :: Segregated Witness Benefits*. 2016. URL: https://bitcoincore.org/en/2016/01/26/segwit-benefits/ (visited on 04/25/2017) (cit. on pp. 16, 45).

[3] *Bitcoin — Open source P2P money*. URL: https://bitcoin.org/en/ (visited on 05/03/2017) (cit. on p. 41).

[4] "Bundesgesetzblatt 2009/123" (2009) (cit. on p. 35).

[5] Vitalik Buterin. *Toward a 12-second Block Time*. 2014. URL: https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/ (visited on 09/01/2016) (cit. on p. 19).

[6] *Developer Documentation - Bitcoin*. 2017. URL: https://bitcoin.org/en/developer-documentation (visited on 03/01/2017) (cit. on pp. 6, 12, 14).

[7] Evan Duffield and Daniel Diaz. "Dash: A Privacy Centric Crypto Currency" (2014). URL: https://www.dash.org/wp-content/uploads/2015/04/Dash-WhitepaperV1.pdf (cit. on p. 42).

[8] Evan Duffield, Holger Schinzel, and Fernando Gutierrez. *Transaction Locking and Masternode Consensus: A Mechanism for Mitigating Double Spending Attacks*. 2014. URL: https://www.dash.org/wp-content/uploads/2014/09/InstantTX.pdf (visited on 05/08/2017) (cit. on p. 42).

[9] *Ethereum Classic*. URL: https://ethereumclassic.github.io/ (visited on 05/03/2017) (cit. on p. 42).

[10] *Ethereum Project*. URL: https://www.ethereum.org/ (visited on 05/03/2017) (cit. on p. 41).

[11] Darya Gudkova et al. "SPAM AND PHISHING IN Q2 2016". *AO Kapersky Lab* (2016), p. 22. URL: https://kasperskycontenthub.com/securelist/files/2016/08/Spam-report_Q2-2016_final_ENG.pdf (cit. on p. 22).

[12] Tony Hansen and Gregory M. Vaudreuil. *Message Disposition Notification*. RFC. RFC Editor, May 2004, p. 29. URL: https://tools.ietf.org/html/rfc3798 (cit. on p. 3).

[13] Ben Laurie and Richard Clayton. *"Proof-of-Work" proves not to work; version 0.2*. 2004. URL: http://www.cl.cam.ac.uk/~rnc1/proofwork2.pdf (cit. on pp. 26, 30).

[14] Charlie Lee. *[ANN] Litecoin - a lite version of Bitcoin. Launched!* 2011. URL: https://bitcointalk.org/index.php?topic=47417.0 (visited on 10/29/2016) (cit. on p. 42).

[15] Thede Loder et al. "The Spam and Attention Bond Mechanism FAQ" (2004). URL: https://www.itu.int/osg/spu/spam/contributions/Spam%20economics-faq.pdf (cit. on pp. 23, 25, 26, 30).

[16] Microsoft. *The Penny Black Project*. 2003. URL: http://research.microsoft.com/en-us/projects/pennyblack/default.aspx (visited on 04/05/2016) (cit. on p. 3).

[17] Keith Moore. *SMTP Service Extension for Delivery Status Notifications*. RFC. RFC Editor, Jan. 2003, p. 38. URL: https://tools.ietf.org/html/rfc3461 (cit. on p. 3).

[18] Satoshi Nakamoto. *Bitcoin 0.3.2 released*. 2010. URL: https://bitcointalk.org/index.php?topic=437 (visited on 03/17/2017) (cit. on p. 19).

[19] Arvind Narayanan et al. *Bitcoin and Cryptocurrency Technologies*. 2016, p. 308 (cit. on p. 40).

[20] *NEM - Distributed Ledger Technology (Blockchain)*. URL: https://www.nem.io/faq.html (visited on 05/03/2017) (cit. on p. 42).

[21] Jack Peterson and Joseph Krug. "Augur: a Decentralized, Open-Source Platform for Prediction Markets" (2015). URL: https://arxiv.org/pdf/1501.01042.pdf (cit. on p. 43).

[22] Joseph Poon and Thaddeus Dryja. "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments". *Technical Report (draft)* (2016), p. 59. URL: https://lightning.network/lightning-network-paper.pdf (cit. on pp. 16, 45).

[23] Justin M Rao and David H Reiley. "The Economics of Spam". *The Journal of Economic Perspectives* 26.3 (2012), pp. 87–110. URL: http://www.davidreiley.com/papers/SpamEconomics.pdf (cit. on p. 22).

[24] *The Golem Project*. 2016. URL: https://golem.network/doc/Golemwhitepaper.pdf (visited on 05/03/2017) (cit. on p. 43).

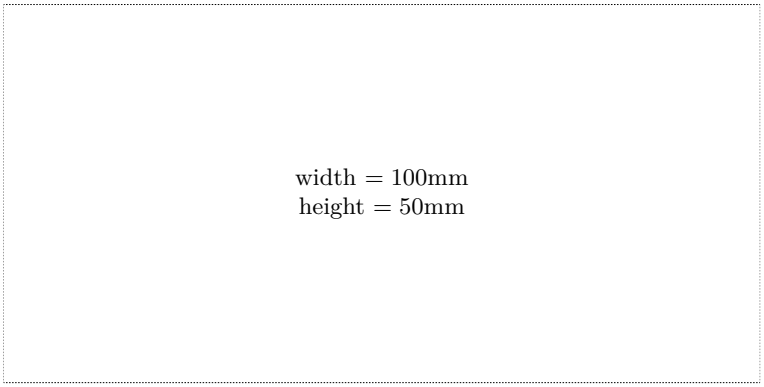
[25] Theymos. *File:Blockchain.svg - Wikimedia Commons*. 2010. URL: https://commons.wikimedia.org/wiki/File:Blockchain.svg (visited on 05/15/2017) (cit. on p. 11).

[26] *Welcome to Ripple / Ripple*. URL: https://ripple.com/ (visited on 05/03/2017) (cit. on p. 41).

[27] *[XMR] Monero - A secure, private, untraceable cryptocurrency*. 2014. URL: https://bitcointalk.org/index.php?topic=583449.0 (visited on 05/08/2017) (cit. on p. 42).

Check Final Print Size

— Check final print size! —



— Remove this page after printing! —