Sending Cryptocurrency over Mobile Applications

Thaís Quintas de Arcanjo

Thesis Submitted for the Award of MSc

School of Computing

Dublin City University

Supervised by Mr. Ray Walshe

January 2019

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Masters of Science is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ____________ **ID No.:** 16212848 **Date:** 3rd of January 2019

Contents

Declaration **I**

List of Figures **V**

List of Tables **VI**

Acronyms **VII**

Abstract **I**

1 Introduction **1**

2 Literature Overview **6**

2.1 Smartphone usage 6

2.1.1 Evolution of mobile messaging 6

2.1.2 Mobile instant messaging 8

2.1.3 Usage 9

2.2 Banking solutions for mobile devices 11

2.2.1 Traditional banking applications 11

2.2.2 Digital alternatives 12
2.3 Payments in messaging applications 13
2.3.1 Mobile electronic wallets for merchant payments 14
2.3.2 E-wallet security 15
2.3.3 Crypto wallet 16
2.4 Bitcoin 19
2.4.1 Validating a transaction 20
2.4.2 Sharing a transaction around the network 21
2.4.3 Transactions 21
2.5 Ethereum 22
2.5.1 Ethereum accounts 23
2.5.2 State, blocks and Blockchain 23
2.5.3 Ether 24
2.5.4 Transactions 24
2.5.5 Messages 25

3 Architecture and implementation **27**
3.1 Structure 27
3.2 Security 29
3.2.1 Securing accounts 29
3.3 Connecting to an existing Telegram account 30
3.4 Connecting to Ethereum 30
3.4.1 Receiving Blockchain information 31

3.4.2 Creating an Ethereum account 32

3.4.3 Getting the account details 33

3.5 Generating events 35

3.5.1 Text messages 35

3.5.2 Transaction messages 36

3.6 Connecting an Ethereum account with a Telegram account . . 37

4 Discussion 43

4.1 Messaging applications viability 43

4.1.1 Data security and development flexibility 44

4.2 Ethereum viability in mobile devices 45

4.2.1 Consumption measurement 45

5 Conclusion 49

5.1 Opinions 50

5.1.1 Ethereum on messengers 50

5.2 Security of Ethereum on mobile device 51

5.3 Future possibilities for messaging applications with cryptocurrency integration 52

List of Figures

2.1 Mobile Instant Message (MIM) Usage [Leung, 2016] 9

2.2 Most popular mobile messaging applications worldwide (in millions of users) [Statista, 2018a] 10

3.1 Application Structure 28

3.2 Execution time 41

4.1 Sidechain Structure 47

List of Tables

2.1 Mobile Application Comparison 18
2.2 Transaction Fields 24
2.3 Message fields 25

Acronyms

PCB	Printed Circuit Board
BHIM	Bharat Interface for Money
UPI	Unified Payments Interface
EKYC	Electronic Know Your Customer
ICO	Initial Coin Offering
OTC	Over The Counter Trading
PoW	Proof-of-workConsensus algorithm to keep track of who owns coins
UTXO	Unspent Transaction Outputs
ISP	Internet Service Providers
MIM	Mobile Instant Messaging
SMS	Short Message Service
IM	Instant Message
GSM	Global System for Mobile
app	Mobile Application
MMS	Multimedia Messaging Service
API	Application Programming Interface
DLT	Distributed Ledger Technology
GB	Gigabytes
QR	Quick Response

Abstract

Thaís Quintas de Arcanjo

Sending Cryptocurrency over Mobile Applications

With the growth of digital assets known as cryptocurrency, where there is no central authority to process a transaction, anyone in the world can participate and start sharing assets with other parties.

Today, chat applications have billions of active users [Statista, 2018a] messaging text, audio, images and video. Transacting with cryptocurrency could be as intuitive and simple as sending a simple message, where users could send and track their cryptocurrency while seamlessly chatting with other users.

Users still need to access third-party services and entrust their credentials to these services in order to have their requests processed. This exposes the users to potentially malicious parties intercepting and manipulating their transactions. If the users were to have control over their assets locally, security flaws would be drastically reduced and transactions simplified on the user's device.

In order to allow users to manage their assets in a more secure way, their data would need to be locally stored in a safer manner such that it would only be accessible by the owner.

The solution is to provide a functional mobile chat application on the user's device that can be used to interact with and manage cryptocurrency using the Ethereum platform. Users do not require any previous knowledge about the platform or cryptocurrencies and this solution will allow them to take advantage of the opportunities that cryptocurrencies afford.

Using the Ethereum platform, the solution application will have its own local node connected to the network, which will allow users to manage accounts, receive and send assets to other users and monitor the status of their assets while chatting.

Chapter 1

Introduction

Communication and payments have been an essential part of human history, and nowadays have come to be an essential service in people's lives. Historically speaking, when nations or certain groups needed to communicate in order to exchange goods or to expand their power and culture, different exchange methods were used in order to conclude a deal or surmount a communication barrier. For instance, to settle a deal, a valuable object could be exchanged. This object, representing a particular value, served as a form of currency that could be used for trading between parties as a payment agreement. In the early ages, such currency objects could vary from pigs and shells to cocoa beans, all of which were considered to be and used as money [Ferguson, 2008]; however, what one group or nation considered to be valuable or hard to forge was not always considered to be as valuable by a different party.

Payment exchanges are not the only thing that has evolved over the years, as so has communication. It has naturally evolved over time and now includes verbal communication forms, such as either spoken or written words, and non-verbal forms, such as gestures, images, songs and more, each satisfying a different need. Due to this communication evolution, technology has had to adapt and has greatly evolved in recent years in order to facilitate person-to-person communication. Years ago, the delivery of letters sent through the post could sometimes take days or even weeks, whereas today various applications enable real-time communication. Consequently, applications capable of exchanging text, audio, image and video files—and more—are central to the lives of millions of people worldwide [Statista, 2018c]. Such messaging applications can connect two or more peers through the internet even if they are at different geographic locations in the world, while providing an easy, quick and interactive conversation service.

Nowadays payment services are more accessible to users because traditional banking service applications have been, allowing banks to offer mobile solution services to their clients and attracting new and younger clientele while also preserving the existing one. The problem with the traditional banking application solutions is that they lack adequate flexibility for integration with non-trusted third-party solutions.

Another example of their limitation is the fact that their payment exchange services are considerably slow, taking days to either be validated or to send information about the outcome of a transaction to both sender and receiver. This is because transaction information is checked and validated by a centralised authority, which is primarily responsible for validating an incoming transaction, thus slowing down the response time from both parties' perspectives. Due to this delay, not knowing if the sender had indeed made the transaction nor when this transaction was completed—and whether it was successful or not—could be frustrating to some users and cause harm to businesses that depend on incoming money in order to begin processing services or goods bought in exchange.

Although it is possible to obtain this information after the central authority has processed the transaction, the problem is that, usually, neither the sender nor the receiver are informed of their account updates and need to manually trigger the update events responsible for collecting this data. This requires the users to open the application and interact with it in order to access and read the latest information available from the central authority's server.

As a way to overcome this issue, a Chinese messaging application called WeChat developed a feature where users could add their credit card information to their account and make payment via Quick Response (QR) code and transfer money from within the app. There were some limitations, however, on the implementation of this service, which until 2018, only accepted credit and debit cards from Chinese banks, and that now, for foreigner bank cards, they only allow using if the application was downloaded from the mainland China, possibly due to security concerns.

Similarly, other bank applications are inflexible and fail in providing feedback over transactions, transparency of transactions and their history. These issues led to the creation of Bitcoin in 2009, which is a digital payment system available in a peer-to-peer network that allows for online transactions between its participants. This system makes use of cryptography (leading to its currency being called a cryptocurrency) to secure and validate transactions in a collective and decentralised manner, which means that all of the transactions are publicly distributed to everyone in the network and all transaction information is made publicly available [Nakamoto, 2009].

This technology is based on Blockchain, a public ledger that is shared among participants (who can join or drop from this ledger at any time) and that depends on its participants to validate all transactions. In order to incentivise users to validate transactions, there is a reward system in place, which rewards participants with coins or fractions of coins based on their contributions to the ledger. If the majority of participants state that participant A, for example, had correctly validated a transaction, then participant A receives a reward for this in the form of coins or fractions of a coin. To identify users on this system, each user has an account represented by two keys—one public and one private. The public key is used to publicly propagate a transaction and the private one is used to sign and lock a transaction. It is only possible to create and sign a transaction using both keys and the private key is never publicly shared on Blockchain, as it represents a critical account security flaw.

After the announcement of this new system and its benefits, the curiosity about and usage of cryptocurrency started to spread and grow among users who adopted this solution, seeking a more transparent, faster and decentralised payment system. As a result of this interest in decentralised payments and their benefits, around 2,500 cryptocurrencies—based on the Blockchain premises—have become available since the initial publication of this system [Investing, 2018]. One of them is Ethereum, which provides a platform capable of running small applications on top of their network platform.

There are several ways to start using cryptocurrency. Some of them require that the user possesses previous knowledge about cryptocurrencies, while others try to hide this complexity from the user to make the process as simple as possible. These solutions are usually server hosted and thus, potentially, lead to major security flaws, with the fact that they usually store the users' account keys on a server (either completely or temporarily) during transaction creation making them potential targets of malicious attacks. This potential threat makes a user's account vulnerable and can lead to a loss of all cryptocurrency available in their account, through a transaction that is signed by someone who happens to obtain the combination of account keys and sends its funds to another account—making this amount gone forever because such a transaction cannot be annulled.

Despite these security problems, the use and number of cryptocurrencies have kept growing and their users have become more conscious, via knowledge sharing in chat text rooms or forums, of the security that such systems demand in their utilisation. Thus, cryptocurrencies are commonly used today as a form of exchange for daily payments. Over 38% of cryptocurrency users have said that they would prefer to make transactions via a mobile device [Incrybit, 2018] and that storing all account keys locally could potentially reduce security problems, allowing users to have control over their account data.

By combining cryptocurrency technology with messaging applications—a form of technology present in most peoples' lives—a better and more intuitive service could be provided for users, who would then be able to make payments while connected to a Blockchain network as easily as they send a simple text message. This combination would differ from the usual ways of dealing with cryptocurrency hosted on external servers, as it would not only be connected to a network and receive all updates from it constantly but, most importantly, a user's sensitive data, such as account keys, would not be transferred to this network nor to other applications running on a user's device. One question, however, remains—would the advantages that cryptocurrencies and their decentralised authority provide be a viable solution for mobile messaging applications?

This research aims to answer this question and to discuss a better approach for managing decentralised payments by comparing the benefits, performance, manageability and maintainability of the source code for instant messaging applications that use Ethereum as their cryptocurrency exchange base.

When discussing cryptocurrency applications, it is essential to consider their storage consumption (as some devices do not have a significant amount of data storage available) as well as the users' inexperience or unfamiliarity with the subject (the application has to be easy enough for beginner users).

Initially, the research covers the evolution of mobile messaging applications and their growth and then analyses mobile payments and their limitations, leading to an examination of cryptocurrencies and a study on the solutions that exist currently. Combining these two topics, a solution architecture is presented and developed while exploring and detailing the concepts of messaging solutions and cryptocurrency. Finally, the developed solution architecture is compared with other third-party solutions.

Chapter 2

Literature Overview

2.1 Smartphone usage

Smartphones are present in most people's lives today since they are easily accessible [Burke, 2016]. The use of a mobile phone in daily life has evolved from a novelty to a necessity, as the traditional mobile communication device has grown into a more robust and powerful substitute. Its existence allows users to communicate with one another instantly in ways distinctive from regular phone communication, such as video calls, connecting people from one end of the world to another.

Mobile phones have evolved from a device capable of sending a text message to one that can manage bank accounts and more. According to a study conducted in 2017, the ownership of mobile devices has increased to the point that 85% of adults in the UK, for example, own a smartphone [Deloitte, 2017].

2.1.1 Evolution of mobile messaging

Approximately 25 years ago, the first Short Message Service (SMS) was sent [BBC, 2002] through a text messaging service available on mobile devices using a standardised communication protocol that allows users to exchange messages. This protocol makes use of an asynchronous communication method, which introduced flexibility to typical voice communication that, until then, was the primary purpose of the telephone, requiring the receiver to be available immediately—otherwise the conversation would be deemed unsuccessful.

This SMS ’text’ technology initially allowed users to send and receive messages up to 160 characters long, person-to-person, and transferred this data across a Global System for Mobile (GSM) network protocol. To send an SMS, a user would pay a mobile provider per message sent. Each message could hold up to 160 characters and if more characters were needed, additional messages were required at additional cost.

As mobile devices became cheaper and more accessible, mobile phones proliferated everywhere, facilitating and increasing SMS usage among users. In 2015, for example, it was reported that the daily average number of sent SMS messages was nearly 7,945 billion [Worldwide SMS Market, 2014].

The usage of SMS demonstrated its popularity but also showed its failings through the lack of support for longer messages and other content types. The need for new services and functionality, demanded by users, propelled further development. This led to the creation of Multimedia Messaging Service (MMS), a protocol that extended the core features of SMS. It did so by allowing messages that were more than 160 characters in length to be sent, as well as pictures, videos, audios and slideshows.

The adoption of MMS was so significant in the United States of America that the monthly messaging traffic reached an average of 18 billion messages in 2010 [Statista, 2013]. This was partly due to the adoption of smartphones, however, one MMS limitation was the fact that not every phone was compatible with this technology. In phones that were not capable of MMS, this type of message would be temporarily stored on a server through a URL and this URL would be sent to the receiver’s phone, whose user would need to open the link via an internet browser. This led to frustration for some users, as an additional payment to the carrier provider was required to open the link.

Although the technologies behind text messaging evolved from SMS to MMS, text messaging was not as easy and convenient as it currently is. Users still had to pay mobile carrier providers to exchange messages and, for a longer conversation, this would be impractical. On the other hand, Instant Message (IM) services that would not limit users kept growing, services such as ICQ, MSN and Yahoo. These messaging applications used the internet as their transport medium, meaning that no extra charges had to be paid in order to send a message. These applications were not only free for data exchange but also had extra features that soon became popular among users, such as the ability to send long messages, provide user status, photo updates, read reminders and more.

IMs were not only cheaper than SMS/MMS but they also revolutionised the way that users conduct online conversations. This service allowed users to send messages to more than one recipient in a single 'thread' and to receive messages on the same basis. Thus, this new service facilitated group messaging functionality.

2.1.2 Mobile instant messaging

Although the usage of IMs was popular, they were not mobile friendly and sometimes had no dedicated mobile application. This obstacle motivated the creation of a mobile application solution, also known as Mobile Instant Messaging (MIM), which was widely used in the 2010s—commonly known as apps such as WhatsApp, WeChat, Facebook Messenger and LINE. These applications were also built on top of the internet protocol and had similar features as those found in IMs but their technology was faster because it was natively developed on top of mobile devices' operating systems.

The advantages of MIMs include the fact that they are more economical and provide instant feedback to users. Depending on the application utilised, a user can know whether the message was broadcast to the intended receiver's device and also whether the receiving user has read this message. Each MIM application usually targets a different kind of user and, in order to attract users, more services have to be provided by these applications. Hence, in some of them, it is possible not only to have group texting but also group voice and video calls.

Nowadays, MIMs are all pervasive for billions of people and it is expected that by 2019 over 2.19 billion people will be using MIMs [MIM Reach, 2015], which creates an increasing demand for current and new messaging applications to offer extra features that will distinguish them from others, for example, by introducing payment methods as a feature.

2.1.3 Usage

Using MIMs also allows for a more virtual social conversation, leading to increased frequency of chatting, planning and group communication [Tang and Hew, 2017]. With all the benefits that these applications have, MIM usage has increased, as seen in Figure 2.1, and the capabilities of these mobile applications have also increased. Today, users can use their smartphones not only to communicate through MIMs but also to play games, work, study, manage their bank accounts and more.

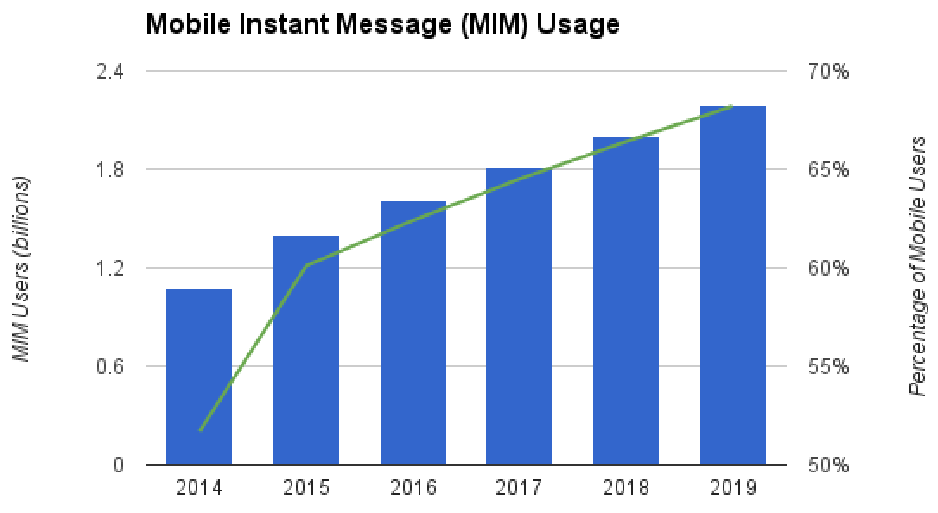

Figure 2.1: Mobile Instant Message (MIM) Usage [Leung, 2016]

The usage of MIMs has spread rapidly among users and, of all the currently available MIM applications, WhatsApp (acquired by Facebook in early 2014) is presently the one with the highest number of active monthly users (figure 2.2), followed by Facebook Messenger and WeChat. It is expected for the usage of mobile messaging applications to go over 2 billion users by 2019.

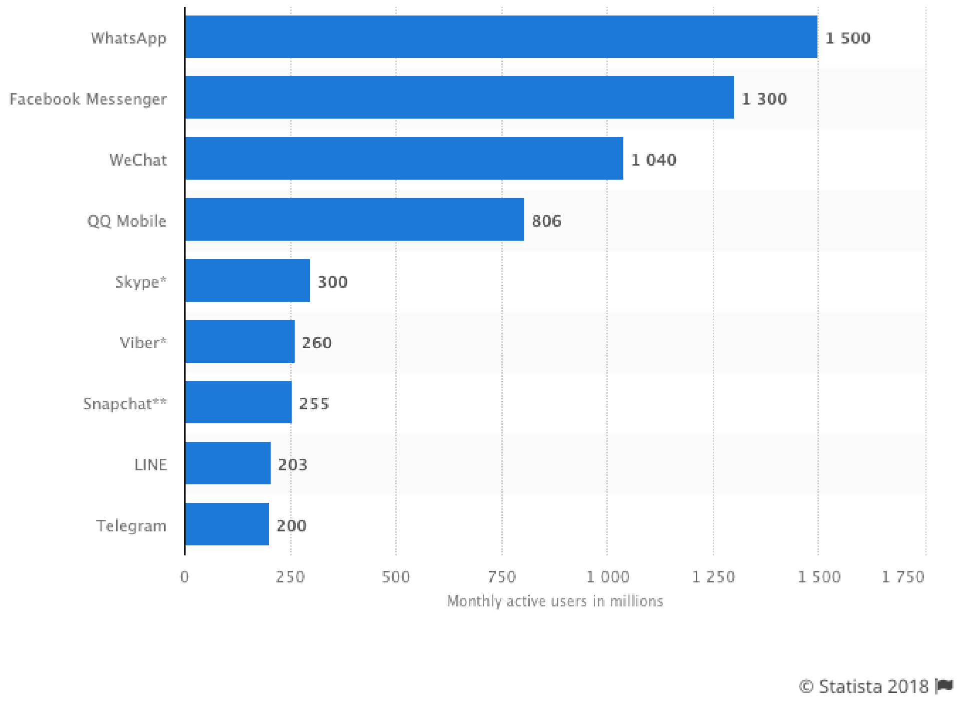

Figure 2.2: Most popular mobile messaging applications worldwide (in millions of users) [Statista, 2018a]

The flow of exchanged messages, for example, is also notable for its large numbers. WhatsApp has nearly 1.5 billion active users and an estimated 60 billion messages are exchanged among its users on a daily basis according to Facebook IQ [Data, 2018].

With all the traffic generated by its application, WhatsApp decided to take a step forward to innovate their messaging application and, earlier this year, it extended their main user-focused messaging application to a business-oriented application called WhatsApp Business. This application allows businesses to develop closer interaction with their clients through normal chat messaging via a BOT-like application.

Facebook Messenger also provides a tool for businesses to interact with their customers inside their application and over 20 billion messages are reported to have been exchanged monthly between clients and businesses within this application [Facebook, 2018].

This trend towards business integration within instant messaging applications is not only for Facebook products only but is also incorporated by WeChat, who developed a business tool for their platform so that users can not only interact with companies but also make payments to businesses within their instant messaging application.

2.2 Banking solutions for mobile devices

When it comes to making mobile payment transactions, currently there are two popular options: Official Bank Applications and Digital Alternatives. Traditional banking institutions usually have official mobile applications available for their clients, while some digital banks integrate the solutions of other banks into their own applications.

It is important to note that an integration between instant messaging applications and banking applications is not always easy to achieve due to security problems. The bank system would need to provide a secure connection to obtain account data, make account changes and accept incoming transactions from third-party systems. Also, the messenger application would need to follow the bank's guidelines about what should be stored on a user's device, which would make the account vulnerable to attacks if not properly implemented by developers. A possible solution for this problem would either be for the bank to own the messenger application (and thus be incorporated into its system) or for the banks to release an interface for their systems to a limited number of trusted third-parties that can hold a virtual reference to a bank account and make payments. This is costly because the company behind the messenger application might incur charges for each transaction even though the application itself is usually free of charge. Adding a transaction fee on top of each transaction solves the cost problem, however, users would still need to trust the messenger company that the sensitive banking data they provide will not be compromised.

2.2.1 Traditional banking applications

Official applications of traditional banks provide consumers with confidence because these banks have been around for decades or even centuries and, as a result of this confidence, the public is naturally more inclined to accept their solutions. Banks are also more comfortable with providing access to Application Programming Interface (API)s, as long as it is from inside the same company, where they have full control over who can access the data and how it is handled internally. This allows the banks to innovate the features they offer inside their applications, as long as their central banking system knows how to handle them. Their clients will eventually try to use these solutions for small and quick transaction processing if it is attractive to use, however, since these applications still rely on the central system, this is not possible unless the central system is updated first.

The confidence is based on the fact that data is encrypted from the time it leaves a device until a response is received by it. Sometimes it is not possible to know whether traditional banking applications implement certain types of data encryption nor whether they store unencrypted data on a user's device. In a big application development project, developers may forget or ignore some standard security procedures, creating what is called a security breach or bug that allows or facilitates malicious attackers to steal the users' sensitive data. As a way to tranquillise customers about these issues, banks may sometimes provide some information about security in their applications, being one of them the data encryption, responsible for securing users' data from the moment it leaves the device until the session is terminated. As this information is not required, it is not possible to state how safe these applications are. Some banks, on the other hand, may provide this information, which is the case of Ulster Bank, one of the largest banks in Ireland, that announced that it uses end-to-end encryption [UlsterBank, 2018] in their application.

Such end-to-end encryption adds another security layer in which the data is encrypted by certain keys that are only known to legitimate parties—i.e. the sender and the receiver—thus reducing the possibility of fraudulent interception attacks, where attackers can read the intercepted information. Another security measure that is implemented in some banking applications is the fingerprint login, where access to the application is granted upon verification of a user's digital fingerprint, thereby reducing the number of possible fraudulent attacks.

2.2.2 Digital alternatives

Internet banking has changed the banking industry, forcing traditional banks to change their business strategies and offer more features to a younger user base. Paypal, for example, could be seen as one of the pioneers of what is known as the Internet Banking (as well as e-banking) system—technically a 'bankless' one—i.e. a system that is not officially a bank. Initially, its primary focus was on providing payment transactions, however, today their business also integrates friend-to-friend payment interactions so that users can send money via their mobile applications from one account to another.

As companies like Paypal became popular due to their easy online payment systems, users also became more open to using non-traditional technologies that make payment transactions more accessible even if it means going 'bankless'. This user acceptance and high level of smartphone usage has also led to the creation of a popular bankless mobile application called Revolut. Revolut is a London based financial technology firm that provides a digital alternative which exclusively targets mobile users—since they do not have a website alternative for the use of their system. This alternative became one of the most used banking alternatives in Ireland in 2018 and has more than 2 million users globally [Cook, 2018]. Within Revolut's application, users can open a new account as well as manage multiple accounts, transferring funds globally in a matter of seconds while only being charged a fee to cover the operation.

2.3 Payments in messaging applications

As users have naturally become more comfortable with using their messaging applications and transferring money through e-banking applications, some messaging applications, such as Facebook Messenger, began to combine the two, allowing users to handle money quickly without having to leave their applications [Facebook, 2015]. A major factor that influenced companies to provide and integrate more services into their solutions was the decline in the number of app downloads [Kafka, 2016] due to the preference of users towards having fewer applications on their devices. Incorporating money transactions into an existing messaging application might help users reduce application storage consumption and encourage users to keep the application on their devices.

Incorporating existing traditional banking applications into their business models is not an easy task for messaging apps because banks have a centralised authority controller responsible for checking the integrity of all incoming access and transactions. Since every transaction must pass through this authority, it makes the process completion time for a transaction longer, sometimes taking days to complete. Integrating such a high latency model within messenger applications, where 'instant' is an expected concept, might jeopardise the product idea. Such extended processing time, third-party access, integration and transactions fees complicate the possibility of banks providing such a facility.

This interest in payment services, from companies such as Facebook and WhatsApp 2.2, demonstrates that the most significant messaging applications are now innovating their platforms to stimulate and attract not only businesses but also their customers for money transactions on top of their applications. This investment in business transactions changes the way technology for messaging applications is perceived. Today, users can already manage their assets through traditional banking applications but this is not as common inside instant messaging applications.

A gap still remains between financial transactions and messaging applications and their integration could be seen as a time-consuming process for users. In order to achieve such integration, for example, logging into the banking application is the first required step, followed by waiting for the latest banking information to become available on the bank's server in terms of balance and account transactions. Afterwards, the user must navigate to the correct screen that allows a new transaction to be made and the receiver's account identification must then be filled in to complete the transaction creation process. This process could be seen as frustrating because both the sender and the receiver could face a delay in receiving confirmation about the transaction that was created for the payment for goods, which could result in a knock-on lag in preparing and shipping orders.

2.3.1 Mobile electronic wallets for merchant payments

Although integration is still a problem, facilitating payments from credit cards is already possible when utilising electronic wallets, or simply e-wallets, which allow payment transactions to be made from a mobile device using an existing card issued by a financial institution (a solution similar to the one that Facebook Messenger offers). This wallet, however, does not hold any value—i.e. its balance is not up to date with a user's bank account—but they do offer an alternative way to make payments from a user's device. Currently, there are several ways to allow such transactions according to the U.S. Payments Forum [Balasubramanian et al., 2018]:

1. Device-centric mobile proximity wallet - This wallet stores the creden-

tials needed in order to make a payment. This wallet accepts cards from any participating issuer and the credentials are issued by the financial institution. The credential can be stored on the user's device or in the cloud.

2. Device-centric mobile in-app wallet - Works similar to device-centric mobile proximity wallet, but it also works with other e-commerce systems that use the tokenized mobile payment in their application, where users authenticate and authorise the payment either through a password or using their biometrics.

3. Card-not-present card-on-file wallet - It is when the user cannot present their physical card at the point-of-sale; this wallet then uses previous credentials stored on the user's device or in the cloud and can make new transactions without user interaction, i.e. repeated or automatic payments.

4. QR code wallet - They are similar to Card-not-present transactions, but they use a QR code in order to complete a purchase.

5. Digital checkout wallet - They also store their credentials in the cloud and can use tokenised payments. The wallet can be accessed via a mobile device or through a website. This creates a system that works similar to a single banking and payment application from the user's perspective.

2.3.2 E-wallet security

While the wallet choices described in 2.3.1 are widely accepted, they should nevertheless be implemented using extra security layers whenever possible in order to protect users against potential fraud. In some transaction settings (e.g. when the card is not present), it is not unusual to have attacks (e.g. data hacking or breach) in which a user's information becomes available without consent through phishing (i.e. when a seemingly legitimate message is sent to a user in order to collect sensitive information) or bot attacks. Some preventive methods might include biometric authentication and device authenticated-only transactions in which a consumer authorises a device before making a transaction and the central authority analyses and blocks any unusual activities from this user.

These methods can be seen in some messaging applications that handle payments, such as 1) WePay from WeChat, where a user can use QR code to make payments at the point-of-sale. In order to use this method, the user should have a wallet containing debit and credit cards, 2) Apple Pay, which is device-centric and only works on Apple devices and 3) Android Pay, which only works on Android devices.

Completely independent from messaging applications, this wallet itself runs separately from an application, which means that the MIM does not have control over the original application. This would not pose a problem as long as the application is an e-commerce application and not a messaging application, where a user sees responses in a matter of seconds (new messages) or even milliseconds (status indicators). In order to make both applications communicate, the user must leave the main application to configure the wallet in an external application and then, after configuring it, return to the caller application that needs to display the original wallet information.

Integrating e-wallets, such as Apple Pay, into messaging applications is still not possible for non-commercial payments, i.e. to a user who is not a merchant. Although the wallet can be configured by and accessed from the messaging application, a user cannot complete the transaction because a unique merchant identifier must be provided, something that does not exist in a one-to-one conversation.

2.3.3 Crypto wallet

A good solution for instant messaging applications would be to have a different type of wallet, which runs independently of any other application but is still flexible enough to have its information accessible and providing account information. Currently, the structure of electronic wallets mentioned previously would not satisfy the needs of an instant messaging application, as integration with an existing banking account would not be feasible—however, with a cryptocurrency wallet it would be.

Cryptocurrency became popular through Bitcoin, which uses a public and secure network that allows anyone to join and validate transactions, i.e. a decentralised authority. This solution, however, is not integrated with existing bank systems and thus requires its own currency, a virtual currency that is usually called a cryptocurrency. This new technology allows anyone, either person or company, to connect and start making transactions with less bureaucracy. One of the first MIM applications to integrate this technology is Kik, a chat messenger with over 300 million users [Livingston, 2016] and the fifth most searched for term in the iOS App Store in 2017. That year the company announced that they would begin using their money transaction service in their application with their own cryptocurrency, Kin, a virtual currency on top of a Blockchain, a Distributed Ledger Technology (DLT) transaction technology that defines decentralised databases without a central authority or intermediator [Mclean and Deane-Johns, 2016] to validate transactions. With this cryptocurrency, they raised almost $100M dollars in a pre-sale round by selling tokens in order to raise money for the Initial Coin Offering (ICO) and product.

Another mobile messaging application, Telegram, also announced that they would begin their own cryptocurrency transactions in their app, one of the top five most downloaded messaging applications in 2018 [Statista, 2018a]. Following the announcement, together with their own ICO, Telegram raised over $1.7 billion dollars in funding, becoming the world's largest fundraising ICO. However, several months later, the company announced that they would no longer pursue that track, without providing details as to why they changed their plan.

A cryptocurrency wallet, or crypto wallet, differs from an e-wallet as it does not possess any information about credit or debit cards or even banking accounts—it primarily contains information about a cryptocurrency account. This cryptocurrency account is protected by two keys: one that is public and available to anyone who wishes to identify the account and one that is private, only available to the creator of the account and used to sign and authorise transactions.

Another way in which these accounts differ from those with a standard centralised authority is that they do not have one entity responsible for ensuring integrity of transactions but have, instead, an open network to which, technically, any device could join, check a transaction integrity and be responsible for signaling all other machines or devices about whether a transaction was valid or not. This network is called Blockchain because transactions are stored in blocks of information that are chained in sequence. Currently, it is estimated that there are over 28 million Blockchain wallets [Statista, 2018b].

This Blockchain network is public and free to use, which means that any device (or any person) joining this network can see all transactions from all accounts. A connection to a Blockchain network is usually provided by its developers or implementers, who provide interfaces for their network and its features for several platforms, such as an Android device, an iOS device, a Windows machine and others. One limitation that is important to note is the amount of information that can be stored on a mobile device. Currently, the full size of a Bitcoin database that holds all transaction history and accounts would exceed hundreds of Gigabytes (GB), whereas most of the current smartphone devices possess less than 128 GB of internal storage.

Current mobile application cases

With the benefits that cryptocurrency use would offer through its various platform compatibility and popularity, some of the most popular MIM applications, as seen in Figure 2.2, have already announced that they would offer in-app integration in order to attract users.

As the research conducted for this thesis aims to integrate a messenger application with a cryptocurrency, the messenger application should potentially have an open API for one-to-one conversations—i.e. an ability to access information from outside the primary application. Some of these applications offer an API layer but, unfortunately, most offer only Bot APIs, a robot that can run automated tasks and send pre-configured messages to users. With a Bot API, it is not possible to get messages from one-to-one conversations but, instead, the call is based on business-client automated conversations, a model inclined to business.

The following table describes some of the nice-to-have features for a messaging application to potentially have for possible external integration:

Table 2.1: Mobile Application Comparison

	WhatsApp	**WeChat**	**LINE**	**Telegram**	**kik**
Bot API	✓	✓	✓	✓	✓
Messaging API	✗	✗	✗	✓	✗
Data Encryption	✓	✗	✓	✓	✗
Payment Methods	✗	✓	✓	✗	✓
Own Cryptocurrency	✗	✗	✓	✗	✓

As seen in Table 2.1, other than Kik, only LINE decided to create a cryptocurrency to be exchanged inside their application and expects to release it in 2019 [Line, 2018]. In order to easily incorporate a new functionality into a messenger application, such as cryptocurrency transactions, the application should be flexible enough to be called or manipulated from external services or applications. This is made possible through open API calls, which allow an external entity to obtain data, listen for new messages and push data (e.g. send a message).

From the messenger applications listed in Table 2.1, only one—Telegram—offers an open API to collect one-to-one messaging conversations while also offering end-to-end data encryption. This application has attracted millions of users, as shown in Figure 2.2, thanks to its secure, end-to-end encryption system and easy integration through its open source API, making Telegram a good option for integration with Blockchain for the purposes of this research.

2.4 Bitcoin

As previously mentioned, cryptocurrencies became popular primarily due to Bitcoin, an electronic cash system publicly announced in 2009 by an unknown person named Satoshi Nakamoto. This system allows for financial transactions to be made without a centralised authority controller that is commonly found and implemented in traditional banking systems. This Bitcoin system is available on a public network, which means that anyone can become a participant, leave the system and join it again at any time as well as check the history of transactions at any point in time.

Since this system is independent from existing monetary currencies, its currencies are commonly called cryptocurrencies because cryptography is used to secure transactions, representing electronic money run by a distributed network of participant nodes, based on transaction agreements. By using cryptocurrency on Blockchain, it is possible to validate that whoever is sending money is not double spending money, i.e. that they are not sending duplicate transactions or spending more than they own.

With Bitcoin, each transaction is a block of digital signatures that is created by hashing previous transactions alongside the public key of the next owner to be added, together with the corresponding amount of transacted coin. However, even though the previous transaction hash is known by the time of this new transaction is created, it is still required to validate that whoever is sending the coin(s) actually owns that amount at that time. In a traditional system with a trusted central authority, this authority checks whether or not the sender owns the amount that is to be sent and this is done entirely privately—only the central authority knows how much money the sender has. However, in order to completely remove this central authority, a new method to validate that the sender can send that amount of coin is required. The solution is to make all participants in a system know how much each of them owns and, to accomplish that, each transaction made in the system is made public, i.e. it is propagated in the network. To propagate a new transaction in the network, system participants first decide whether a transaction is valid or not by having the majority of the nodes (i.e. the participants) agree on the validity of the proof that the creator of the transaction (the sender) has sent. Once the majority of the nodes acknowledge it as a valid transaction, the transaction is then added to a single transactions history tree.

2.4.1 Validating a transaction

All coins in this system that are not spent are located in a place commonly called 'state'. This state also has the owner's public key address. Whenever a new transaction is created and inserted into the history tree, a validation is made in order to secure the integrity of the new node. This validation is called a timestamp server validation and consists of creating a specific hash of the block of items, timestamping and publishing it; this ensures that the specific data did exist in the account at the time of the transaction.

The timestamp generates a new hash value that is added to the transaction block. Every timestamp includes the previous timestamp in its hash, thus making a chain in which the next node always knows the previous node's timestamp, thereby increasing its integrity. By using this timestamp hash, it is possible to prevent senders from spending currency that does not exist as well as to prevent senders from trying to use currency that belongs to another participant.

In order to implement and distribute a timestamped server on a peer-to-peer basis, it is necessary to use a system that deters attacks which try to insert invalid nodes or to manipulate previously validated nodes into the history tree. Such a system is called proof-of-work (PoW) and its goal is to make it more difficult for an intruder to manipulate it while, at the same time, making it easy for honest nodes to verify their legitimacy. With cryptocurrency, the proof-of-work system basically involves ensuring that a new hash begins with a certain amount of zero number bits. Once this requirement is satisfied, the block cannot be changed without recalculating the computation to generate a new hash.

2.4.2 Sharing a transaction around the network

When a participant tries to make a new transaction, the process to authenticate and propagate this new transaction is that all the participants connected to the network receive this new transaction and put it in a block to find the difficult proof-of-work for the newly created block that holds the transaction. When the first participant finds a legitimate value, it then proceeds to propagate around the network its new data; however, even though all nodes receive the new block that looks legitimate, all nodes that received the block make another validation to ensure that all transactions in the block are valid and that they have not been spent before. After other nodes accept this block as being valid, each node then proceeds to add the new block to the current chain of blocks.

When more than one node propagates a new block into the network at the same time, all the other receiver nodes may receive them in a different order, and this might cause a difference in the network of nodes. In order to overcome this issue, all nodes consider the longest chain of nodes as being the correct one and when blocks are propagated simultaneously, each receiver node will consider the new block as the valid next block but will also keep a copy of the other possible chain with the other block broadcasted, creating a tie of chains. This tie is broken once the next proof-of-work is found and the block is propagated around the network making a specific branch longer, which makes all the other nodes that were working on the other branch switch to the longer one.

2.4.3 Transactions

To send money between peers, the nodes, also called miners, who find the proof-of-work charge a specific value for the time taken to compute it; this value is called a gas value. Each computation made by a node is called a unit of gas and each unit might have a different price. When a participant creates a new transaction, they must inform how much they are willing to pay for each unit of gas spent by the node that found the proof-of-work and each unit of gas is multiplied by the value of gas defined by a participant; thus, money is one way to persuade participants to run these computations.

In Bitcoin, a transaction might take a few minutes to be computed and added to the Blockchain. This delay is caused by the blocktime that is set to 10 minutes. This blocktime limits the number of transactions that each block might contain so that a transaction could wait for several blocks to be confirmed. It is possible to have this transaction confirmed sooner by paying a higher gas value to have it prioritised by nodes responsible for running the computations.

Certain problems related to the Bitcoin currency, such as the cost issue and its lack of flexibility over conditions on transactions, led to the creation of other cryptocurrencies, commonly called altcoins. While these altcoins based their cores on Bitcoin, they also explored some of the disadvantages that this currency system had in order to create new cryptocurrencies that addressed particular issues. One of these altcoins is Ethereum, which addressed the Bitcoin blocktime issue that led to higher gas values by not allowing a new block to be added every 10 to 20 seconds. Hence, the integration of the Bitcoin system into an instant messenger application might frustrate users due to its lengthy confirmation time, making Ethereum a better choice because of its faster response time.

2.5 Ethereum

Like Bitcoin, Ethereum is also a decentralised platform/protocol of cryptocurrency based on the blockchain network and on the use of tokens that can be bought, sold or traded through smart contracts. Ethereum provides several features on its platform by using a programming language that can create logical rules, called smart contracts, which could be used to enforce conditions and rules for transactions. This platform also provides a way to introduce a custom currency into the system.

Similar to Bitcoin, the Ethereum network is a collection of participant nodes that are all connected to other nodes in the same network and that can, individually, execute programs, i.e. independently from other nodes. There are several networks available for Ethereum—the Main, Test and Private networks—and each of them targets a specific purpose.

The Main network is available for any node to connect to and its currency value is agreed to be worth real money. The Test network allows developers to develop applications on top of Ethereum and the currency exchanged in this network does not hold any value in the real world—which fits the purpose of this research. The Private networks are available for anyone to create, however, in order to join, the person who created this private network must share their settings and location with other nodes in order them to be able to connect to their network. Private Networks can be reset at any time.

2.5.1 Ethereum accounts

The state is made up of objects called 'accounts'. Each account is identified by a 20-byte address and contains the following fields:

- The nonce, which is a counter to ensure each transaction is only processed once
- The account's current ether balance
- The account's contract code if applicable
- The account's storage location, which is empty by default

Two account types exist in Ethereum: an externally owned account that is controlled by a private key (alongside a public key) and a contract account that is controlled by a contract code. The difference between the two accounts would be like comparing a person with an autonomous agent. The person would only be able to make transactions, while the agent would only work when requested to do some job. This agent could also manipulate the storage with newly reported information as well as contact other people or contracts created in the network.

2.5.2 State, blocks and Blockchain

According to the official White Paper of Ethereum, a state is the collection of all accounts ever created in Ethereum and their contents. A block, on the other hand, is a uniquely identified object that holds the transactions and the current state at the moment of the new block creation. Each block is subjected to validation in order to be stored in the state, as seen in section 2.4.2. After the verification and validation of a block, the current Blockchain state should be updated to match the latest block information—i.e. how each account should be updated—and this state would then become the latest state in the Blockchain. In order to preserve the history of all blocks, a chain of blocks is stored and this is called a Blockchain.

2.5.3 Ether

Ether is the main unit of value in the Ethereum system and it is also used to pay the transaction fees. There are several division units for a single ether and each unit gives a sub precision rather than a long number. Ether is also the name of the currency value for Ethereum, representing 10^{18} of Wei, the smallest base unit.

2.5.4 Transactions

A transaction in Ethereum is a signed data package that holds the message to be sent from an externally owned account. This table is created from an actual user account and it holds the following data within:

Table 2.2: Transaction Fields

Name	**Description**
*Receiver's key	The account public key of who is going to receive the message.
*Sender's key	A unique identifier of the sender.
*Amount of Ether	The amount of ether that the sender is transferring to a receiver.
Start Gas	Maximum computational execution value that the transaction could make.
Gas Price	The value of how much the sender is willing to pay for each computation execution.

* Mandatory fields

The optional fields, Start Gas and Gas Price, could be placed as an exit if the contract code is not being exited due to an intentional or accidental infinite loop or to poor code that consumes too many resources from a contract, which would make the transaction too expensive for the sender. Each unit of computational execution inside a function is referred as a 'step' and each step has its 'gas' price defined. The minimum gas price for all steps is 1 but some computation operations might cost more, as they could be more computationally expensive to the node that is validating the transaction.

2.5.5 Messages

A message is an object similar to a transaction object but, instead of being created by a participant in the network, all messages are only created by the contracts—i.e. they are not serialised and only exist at the time that a contract code is called. A contract can send messages to other contracts deployed in the network. A message object contains the following fields:

Table 2.3: Message fields

Name	**Description**
*Sender's Public Key	The account public key of who is going to receive the message
*Receiver's Public Key	An unique identifier of the sender.
*Amount of Ether	The amount of ether that the sender is transferring to a receiver account.
Data	Optional data field to be sent.
Start Gas	Value that specifies the maximum computational execution that the transaction could make.

* Mandatory fields

This chapter shows that, as mobile usage grows, it transforms the way in which users run their daily tasks, with many running them nowdays from a mobile device connected to the internet. Messaging was one of the tasks that had transformed the most with the development of instant messaging applications, which have transformed from a simple application able to send and receive text messages only to a more sophisticated application capable of handling multiple types of conversation. Today, messaging applications continue to slowly expand their capabilities to include tha ability to make monetary transactions. This expansion, however, is usually more focused on businesses rather than individuals. This could, potentially, change through the usage Ethereum, which could allow regular individuals to handle transactions inside any application even a messaging one.

Chapter 3

Architecture and implementation

As described in chapter 2, Blockchain and Ethereum are flexible enough to be incorporated into some messaging platforms and devices through the open source libraries they possess that are available for multiple platforms. For a feature to be embedded into an official messaging application, it should be flexible enough to allow customised views to the application's users, which is not currently available for the majority of the most used messaging applications.

In order to allow a user to connect to an existing wallet, a new application would need to be developed that would connect the two systems: the MIM and the Ethereum. Currently, one of the most open MIM applications on the market is Telegram, which has all of its APIs made available to the public—i.e. it is ready to use and documented. Not only are the services open source but so is its client application; that is, the official application available in Apple Store and Google Play is officially available for download from its creators, which means that it is possible to download the source code for this client application easily. With this in mind, the solution application developed for this research uses the official Android source code from Telegram [Telegram, 2018] and has a local wallet running on a user's device.

3.1 Structure

As the application starts up, it first needs to connect to the Blockchain before the user can interact with it. In order to do so, the application will need to:

- Connect to a network
- Start a local node and create a Keystore to hold all protected accounts
- Connect to the node from a Provider instance
- Download headers for Blockchain to keep the node up-to-date with the network
- Connect to a public node to ask for the validation of blocks

The solution system implements a bridge between the Ethereum network and the local wallet, which is running separately. The bridge should be instantiated by the application when it is initialised. By connecting to a node, the data from the latest available Ethereum state can be accessed, which means that users can read their accounts' balance and also receive information about new transactions added to the Blockchain. This connection ensures that the user is always up to date with the main Ethereum tree. The final structure can be seen in Figure 3.1.

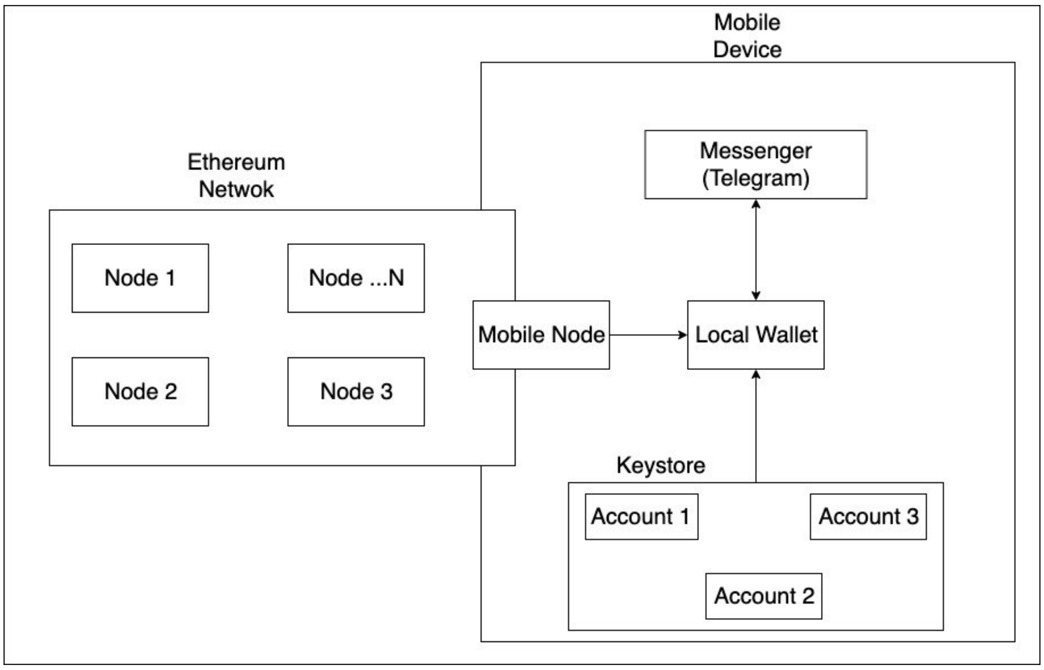

Figure 3.1: Application Structure

3.2 Security

As accounts are stored and can be accessed by knowing the private and public keys, storing the account keys on a public server or a centralised database is not recommended by Ethereum. Instead, the user is responsible for storing this sensitive data in a secure place at their own risk.

After the application creates a node and connects it to the network, all accounts and keys are stored on the device and are going to be valid for access until the user decides to remove this data from the device. As a result, the client will handle the accounts only from within the application, restricting access permission to other nodes, as well as the user's, Internet Service Providers (ISP) because the data is not transmitted in the network. While access is limited by never transmitting the user's account keys around the network, other applications running on the user's mobile device could still have access to this sensitive data. In order to secure authorised access, all accounts are encrypted in a Keystore folder location which, by default, is inside the application data directly.

Although the Keystore location is not easily accessible by other applications, in a rooted device, it would still be possible for another external application to have access to the accounts stored in the Keystore location. Such an external application would still require a password to unlock all accounts in the Keystore and, as each account is individually encrypted by a passphrase provided by the user when creating a new account, this provides a further security measure for credential separation according to Ethereum.

3.2.1 Securing accounts

In a light client, an experimental feature by Ethereum, only the latest block and state get stored in the user's device, which means that it is only possible to find a transaction by hash locally if the transaction does exist locally—other requests can be made on demand. The application can still connect to an external server and request the transaction by its hash, receive the information back from the server and check the veracity of the answer by also requesting the transaction from several other servers for confirmation.

As accounts are individually encrypted and stored in the Keystore, the application must authenticate the account from which the transaction is coming through a passphrase for each signed transaction. This means that, when creating a transaction, the passphrase for the account from which the transaction is coming must be provided by the user in order to decrypt the private key for that account. This decrypted private key is automatically thrown away once the transaction is signed.

It would, however, be possible to cache the user's passphrase, unlock all accounts that the user has stored on the device as long as the application is running and, when the user decides to make a new transaction, to use the unlocked account's public and private keys to sign a transaction. However, if the user leaves the mobile device unattended or a malicious application is running on it while accounts are unlocked, then it would be possible for anyone to make a new transaction because an unlocked account does not require the passphrase to make a new transaction.

3.3 Connecting to an existing Telegram account

As discussed before in section 2.3.3, the application is going to be built on top of the existing official Telegram client for Android.

A requirement to start the application is to provide a unique application ID and hash provided by Telegram itself through the official Developer Section available on their website. With that ID, it would be possible to authorise the application and eventually use it to connect to Telegram on the user's device.

3.4 Connecting to Ethereum

In order to connect to Ethereum, the official implementation of the Ethereum protocol, which supplies an entry point into the Ethereum network, is used because it also provides some helper classes utilised in order to connect to a network and store accounts.

When connecting to an Ethereum network, the application must provide some information about where the Blockchain headers are going to be stored on the user's device as well some configuration aspects for the Ethereum network the user is connected to. With this pre-configuration that the node requires, the node can then be started by calling the method ’start’, as seen in code snippet 01.

```
private Node initialiseEthereum() {
    NodeConfig networkConfig = new NodeConfig();
    String basePath = getReactApplicationContext().getFilesDir()
        + "/";
    String ethereumStoragePath = basePath + ".ethereum";
    String keyStorePath = basePath + "keystore";
    this.keyStore = new KeyStore(keyStorePath, LightScryptN,
        LightScryptP);
    Node node Geth.newNode(ethereumStoragePath, networkConfig);
    node.start();

    return node;
}
```

Listing 3.1: Code Snippet 01: Connecting to the Blockchain

The Keystore could also be initialised when starting the node and the application then specifies the location in which the keystore shall store the account’s data, as well as how resource intensive the encryption used by the keystore should be. The parameters LightScryptN and LightScryptP are integer values that represent the parameters N and P of the Scrypt encryption algorithm, respectively, with both consuming approximately 4 MB of memory and taking about 100 ms of CPU time to be executed on a modern processor. Those values are smaller in comparison to the standard encryption algorithm that consumes 256 MB of memory and approximately 1 s of CPU time on a modern processor, benefiting mobile applications with its small storage and CPU consumption.

3.4.1 Receiving Blockchain information

By using a light client, the application will only receive headers that hold information regarding the latest block added to the chain, such as the latest block number, hash, parent hash, the gas used and so on.

To synchronise the headers on the network through the connected node returned from the Ethereum initialisation method, we will utilise the Ethereum client running inside the local node that works as a facade for accessing some information pertinent to the Blockchain.

```
private void synchroniseHeaders(Node node) {
    Context context = new Context();
    Long bufferSize = 16;
    EthereumClient client = node.getEthereumClient();
    client.subscribeNewHead(context, getNewHeader(), bufferSize)
        ;
}
private NewHeadHandler getNewHeader() {
   return new NewHeadHandler() {
       @Override
       public void onError(String s) {
           Log.e("ERROR", s);
       }
       @Override
       public void onNewHead(Header header) {
           Int blockNumber = header.getNumber();
           String headerHashInHex = header.getHash().getHex();
       }
    };
}
```

Listing 3.2: Code Snippet 02: Synchronising data

As there is only one client per executed node, a listener is created that should be called whenever a new header is received or if there was an error while downloading or verifying a block of headers. This listener will be passed to a subscriber method together with the context, which is the scope for requests that can be cancelled if there is an error or that can otherwise collect information of the received request response in a buffer of 16MB.

The header received from the method 'onNewHead' could then be stored separately (it is already stored locally on a storage path specified in code snippet 01) or even be displayed to the user on an informative status screen, for example.

3.4.2 Creating an Ethereum account

In order to transfer ether, the user needs to have an account to indicate where the money is coming from as well as a destination account address. In Ethereum, there can be two different accounts—a standard account and a contract account. In order to allow the user to take the private and public keys for a standard account and use them inside the app, a standard account is used. A passphrase would restrict the access to this account and therefore, when creating a new account, the user must provide a passphrase that will be attached to the keystore with the newly created account.

There is no limit as to how many accounts a user could create, as a Keystore stores a wide array of accounts. The Keystore will always point to the same directory in which the encrypted accounts are stored and will be responsible for loading them. As can be seen in code snippet 03, the instance of a Keystore is responsible for handling the creation of accounts and the user must specify a passphrase when requesting this functionality. In case that an invalid passphrase is provided by the user, whether this is undefined or empty, the Keystore will produce an error message that the user must supply a valid passphrase.

```
public void createAccount(String passphrase) throws Exception {
   Account newAccount = this.keyStore.newAccount(passphrase);
}
```

Listing 3.3: Code Snippet 03: Creating an account

3.4.3 Getting the account details

All accounts that a user creates locally by the Keystore do not hold any information about the state of the account—i.e. they do not know how much ether that account holds, they only store the public and private keys of an account. Therefore, in order to access the balance of all accounts stored locally, the balance of each account must be requested via the current Ethereum client based on the current header received from the node, i.e. the latest header information about the Blockchain that the node has received.

```java
public void getAccounts() throws Exception{
    final Accounts accounts = this.keyStore.getAccounts();
    ArrayList<String> accountsArray = new ArrayList<>();
    int blockNumber = -1; // latest block number known


    for (long i = 0; i < accounts.size(); i ++) {
     Account account = accounts.get(i).getAddress();
␣ Context context = new Context();
␣ EthereumClient client = node.getEthereumClient();
       final BigInt balanceAt = client.getBalanceAt(context,
          account,           blockNumber);
   }
}
```

Listing 3.4: Code Snippet 04: Getting account details

As can be seen in the code snippet 04, the method "getAccounts" available from the KeyStore instance will return an array of accounts, and we could loop through all accounts to access and get all their public addresses from the "getAddress" method. With the account address, the balance for that account is requested through the client, and the block number that the requested data information should come from is specified, i.e., from which block inside the Blockchain, that represents a specific time, where the request should return data. By passing "-1" as a parameter to request, it states that the balance for that account should come from the latest block number inside the current header received inside the node. This latest block number could also be taken from the header received on the listener available on code snippet 02.

As can be seen in code snippet 04, the method 'getAccounts', available from the KeyStore instance, will return an array of accounts and we could loop through all accounts to access and obtain their public addresses using the 'getAddress' method. With the account address, the balance for that account is requested through the client and the block number that the requested data information should come from is specified (i.e. from which block inside the Blockchain, that represents a specific time, should the request return the data). By passing '-1' as a request parameter, it states that the balance for that account should come from the latest block number inside the current header received inside the node. This latest block number could also be taken from the header received on the listener, available in code snippet 02.

3.5 Generating events

In messaging apps, both devices must connect to the same server in order to fetch new or past events, which could be new messages, transaction information, typing events, multimedia data and more.

In case the user is not connected to the internet and is thus unable to receive new incoming data, the server should also store events' information which should only be accessed by relevant parties.

Every plain text message and transaction attempt needs to be stored to allow a user to read their history. For plain text messages, only three pieces of information are required to be stored in the server's database: the sender's and the receiver's unique identification and the content of the message. Although all transactions are publicly stored on the network, it is not efficient to request and read all the network blocks to find transactions that match the criteria—for example, to find transactions from a specific sender or receiver—the key transaction information should still be stored on the server for faster access. The full transaction details could be requested from externally running full nodes, as they store all the block information.

3.5.1 Text messages

Through the official Telegram API, it is possible to send a message to a user by providing the user's unique identification, the text itself and a random identification number required in order to prevent the re-sending of duplicate messages. This random ID is just a long number that represents a 64-bit signed number.

```java
public void sendTextMessageToUser(String message, Long toUserID)
    {
    MessageObject replyingMessageObject = null;
    TLRPC.WebPage messageWebPage = null;
    boolean messageWebPageSearch = false;
    ArrayList<TLRPC.MessageEntity> entities = new ArrayList<>();
    TLRPC.ReplyMarkup replyMarkup = null;
    HashMap<String, String> params = null;
    SendMessagesHelper.getInstance(currentAccount).sendMessage(
        message, toUserID, replyingMessageObject, messageWebPage,
         messageWebPageSearch, entities, replyMarkup, params);
}
```

Listing 3.5: Code Snippet 05: Sending text message between accounts

Code snippet 05 would then inform the API of a new message to a user, while the API itself would generate a random ID to be attached to the original request after preparing the request object. In order to listen to the status of the send message, this information would be received asynchronously, i.e. at any point in time, and would be received from the server, notifying whether the message was successfully sent or, if an error occurred, specifying the error message.

3.5.2 Transaction messages

Every time a user creates and sends a transaction on the Ethereum network, a transaction receipt is created in order to hold information regarding the results of its execution. This receipt is an object that contains the status of a transaction, whether it has failed or succeeded, the transaction hash as well as the logs from its execution. With this, even if the transaction is rejected, its rejection and cause are still accessible because it is attached to this object and stored in the network state in full running nodes. When the transaction is found valid and the new block holding the transaction is added to the chain, the receipt holds all the same information as a rejected transaction but its status is a valid one. All transactions require a valid private key to be signed into them and this information is provided via the user inputting the account passphrase. This passphrase is used to encrypt the user transaction and this transaction is dispatched in a new action from the running node.

```java
public String createTransaction(String accountHash, String
    passphrase, String accountToHash, long gasLimit, BigInt
    amount, BigInt gasPrice, String data) throws Exception {
    BigInt NETWORK_ID = new BigInt(4);
    Account from = getAccountByAddress(accountHash);
    Address to = new Address(accountToHash);
    Context ctx = new Context();
    long nonce = client.getPendingNonceAt(ctx, from.getAddress()
        );
    final byte[] dataBytes = data.getBytes("UTF8");
    Transaction transaction = new Transaction(nonce, to, amount,
         gasLimit, gasPrice, dataBytes);

    Transaction signed = this.keyStore.signTxPassphrase(from,
        passphrase, transaction, NETWORK_ID);
    client.sendTransaction(ctx, signed);
    return transaction.toString();
}

private Account getAccountByAddress(String address) throws
    Exception {
    final Accounts accounts = this.keyStore.getAccounts();
    for (long i = 0; i < accounts.size(); i ++) {
        if (accounts.get(i).getAddress().getHex().equals(address
            )) {
            return accounts.get(i);
        }
    }
    return null;
}
```

Listing 3.6: Code Snippet 06: Sending ether between accounts

3.6 Connecting an Ethereum account with a Telegram account

Now that the messaging application is running, it has acquired its credentials and the new crypto wallet is dynamically created and allocated by the application so that any part of the application accessible after this initialisation can request data from the crypto wallet and the node. The next required phase would be to connect the user's Ethereum account with the user's messaging account. Since some smartphones users may not lock their devices when they are not in their possession, in order to minimise frauds and for security reasons, the application should only allow a transaction request to be sent after confirming the user's identity. This could be achieved by checking whether the user's device has a biometric reader, as well as by requiring a passphrase that would be used to unlock the wallet, or by creating a password for the user's wallet before proceeding to create a transaction.

In order to connect to the user's account, the application could read all available accounts on the user's local wallet and display them in the application. Information is displayed on the user's messenger account information view inside the application to improve usability and to keep any user-specific information on a single screen.

A disadvantage to using an existing open source API for a messaging application is not having the flexibility to append new user information to the existing API model. On this application, the only information to identify an existing user with an existing Ethereum account is their public key. The current Telegram API provides the user's ID; however, it does not allow for the appending of extra information, such as the Ethereum account, In order to do this, the user would need to make this information explicitly visible in their account (e.g. name) or to provide it via a message chat. This information would be publicly available to everyone, even though it might not be in the user's best interests to have this information announced to non-personal contacts. Therefore, in the application developed here, whenever sending a user's Ethereum public key is required, the receiver should provide it through a message before the transaction creation event is sent to Ethereum.

It is essential for this application not to display too much information at once for the user, considering that many users may have limited knowledge of cryptocurrency. The application should be intuitive; otherwise, users may find the application too complicated, which would result in limited adoption.

To facilitate increased adoption and usage, the application should not overload the screen with too much information when making a new transaction but should simplify its usage instead. The user should see sending cryptocurrency transactions to be something as simple as sending a photo or file. By default, Telegram offers a button that toggles the type of message being sent, varying from photos to files, location and more. The solution application appends an extra button option to include cryptocurrency transactions. After clicking this button, the user would need to provide the following information required for the new transaction:

1. The public key of the sender's account

2. The public key of the receiver's account
3. The amount of ether
4. The passphrase to unlock the account
5. The gas limit
6. The gas price
7. Optional extra text data of the transaction

To avoid overloading the screen and the user with too much information at once on the same chat screen, the user could be redirected to another screen responsible for collecting all the required transaction information in a user-friendly manner. Since all accounts are stored locally, the application will autofill this information with the user's account key, adopting a similar approach as defined in section 3.4.3 facilitating and speeding up the transaction creation time.

The application could improve the usability for the user by providing the gas limit and gas price entry values, as these values could change daily. For the gas price, Ethereum provides a helping method to return the suggested gas price based on recent blocks added to the network using the 'suggestGasPrice' method. The gas limit could be suggested by retrieving the gas limit used on the latest block found, as can be seen in code snippet 07.

```
private void suggestValues() throws Exception {
    final long gasLimitOfRecentBlock = client.getBlockByNumber(
        context, -1).getGasLimit();
    String suggestedGasLimit = String.valueOf(
        gasLimitOfRecentBlock);
    String suggestedGasPrice = String.valueOf(client.
        suggestGasPrice(context));

    ((EditText)findViewById(R.id.gasLimit)).setText(
        suggestedGasLimit);
    ((EditText)findViewById(R.id.gasPrice)).setText(
        suggestedGasPrice);
}
```

Listing 3.7: Code Snippet 07: Pre-populating gas price and gas limit

By asking the user for the passphrase—which is never stored by the application—the application reduces potential malicious attacks in cases where the user did not initiate a transaction.

After the user provides all required information, the application then performs a second check to ask the user whether it should proceed to create a new transaction using a yes or no question in order to proceed with the process. If the user decides to cancel the transaction, all information regarding this transaction should be deleted from the application state; otherwise, the application should use the method called 'createTransaction', specified in section 3.5.2, passing as a parameter the provided information from the user.

```java
private void sendTransactionMessageToUser(String extraData, Long
     receiverTelegramID) throws Exception {
    String toAccount = extractText(R.id.toAccountHash);
    String inputedGasPrice = extractText(R.id.gasPrice);
    String inputedGasLimit = extractText(R.id.gasLimit);
    BigInt etherAmount = extractBigInt(R.id.ether);
    BigInt gasPrice = client.suggestGasPrice(context);
    long gasLimit = client.getBlockByNumber(context, -1).
        getGasLimit();
    if (!inputedGasLimit.isEmpty()) {
        gasLimit = Long.valueOf(inputedGasLimit);
    }
    if (!inputedGasPrice.isEmpty()) {
        gasPrice = new BigInt(Long.valueOf(inputedGasPrice));
    }
    String fromAccount = extractText(R.id.fromAccountHash);
    EditText passphraseComponent = findViewById(R.id.passphrase)
        ;
    String passphrase = passphraseComponent.getText().toString()
        ;

    try {
        String transactionID = this.createTransaction(
            fromAccount, passphrase, toAccount, gasLimit,
            etherAmount, gasPrice, extraData);
        this.sendTextMessageToUser(String.format("Sent %s to %s.
             Transaction ID is %s.", etherAmount, toAccount,
            transactionID), receiverTelegramID);
    } catch(Exception e) {
        this.sendTextMessageToUser("Could not complete
            transaction:" + e.getMessage(), receiverTelegramID);
    } finally {
        passphraseComponent.setText("");
    }
}
```

Listing 3.8: Code Snippet 08: Providing transaction information

After the transaction is sent, the user then receives the transaction hash from the response of the method call. This hash gets sent as a text message, defined in section 3.5.1, to inform both users that this transaction was created and can also be tracked outside the application. Since the application is already checking for new blocks added to the main branch, as defined in section 3.4.1, as soon as a block containing the same transaction hash from the section 3.5.2 is identified, the application can automatically notify both users about the outcome of this transaction.

The execution time for the 'sendTransactionMessageToUser' method, defined in code snippet 08, may vary depending on the user's internet connection speed. As the method is responsible for creating a transaction and sending it out to be validated, on a 4G network connected to a test network, it could average 1.05 s, as can be seen in Figure 3.2.

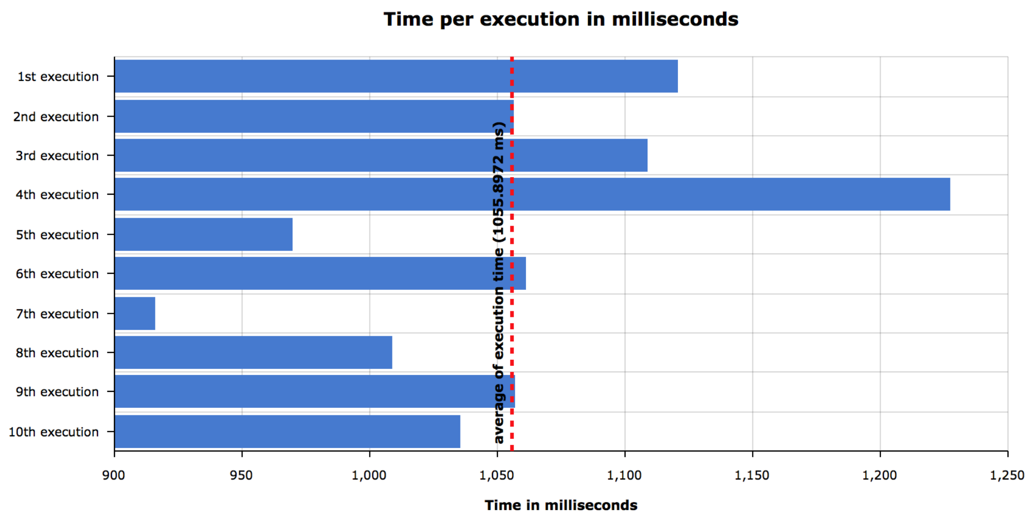

Figure 3.2: Execution time

Although sending a transaction might not take long, as section 2.4.3 shows, a new block is only added to the main chain every 10 to 20 seconds, which would then be received from method 3.4.1.

As seen in this chapter, integrating Ethereum with a mobile device and using it inside a chat application is not overly complicated is using the open sources libraries created by the Ethereum team and MIM application providers, such as Telegram. By providing a library that creates a light node, Ethereum supports users who may not want to compromise their storage allocation on a device, which is a necessary feature for a mobile device. Since not every popular messaging app is as open source as Telegram is, this limits the possibility of integration and forces users out of their current application to fulfil transactions through an external service provider, which is subject to external attacks.

From a user perspective, the total execution time of a transaction request is not very long, as previously demonstrated, and simple suggestions can be made to users to teach them about the data whose value changes daily, such as the gas price and limit values.

Chapter 4

Discussion

As demonstrated in chapter 3, it is possible to combine cryptocurrency technology with instant messaging applications to facilitate secure payment transactions and its incorporation into messaging applications could be quite naturally seen as a new type of messaging. Although the cryptocurrency world may be unpredictable, its current state of development allows for flexibility in the implementation of its core capabilities into several other platforms, including mobile devices.

By using the Ethereum Geth library and running it from within a messaging application, in this case Telegram, it is possible to connect to a Blockchain in the background and to allow users to perform transactions while chatting to other users. This solution could be used by millions of current wallet holders and would also serve to educate people—who often have to rely on third-party providers to handle transactions instead of having the capability to control them locally—about the cryptocurrency world.

4.1 Messaging applications viability

In order to integrate different technology sectors, such as cryptocurrencies and messaging applications, the holder application that would be in control of triggering actions should be flexible enough for developers to implement solutions on top of their existing technology. An ideal messenger application should be able to altogether provide, using end-to-end data encryption, one of the following:

- Provide external access to user API
- Provide source code to integrate requests into their main source base
- Provide updated documentation

4.1.1 Data security and development flexibility

As can be seen in section 2.3, end-to-end data encryption is present in the most popular mobile applications. Telegram is currently one of the most flexible options, as their application not only incorporates end-to-end encryption into their solution but also ensures that third-party requests conform by releasing their official application on Play Store via copyright software license, demonstrating confidence in their security layer even if the requests are not from their official solutions. This level of security not only attracts a new user base but also gives existing users more confidence in their information and who has access to it.

Apart from that, their robust open-source API allows developers to integrate their solutions into any application, even non-chat applications; for example, a bot that is responsible for handling product enquiries. This API allows a level of flexibility that is not usually available from the top downloaded applications. In an ideal scenario, every messaging application would offer a full open-source API for integration. This has already started to happen, as can be seen with WhatsApp, which now offers an API dedicated for bot applications but not yet for user applications.

Telegram, on the other hand, although possessing flexibility through their API, still needs to provide more detail in their official documentation regarding the interaction and required steps for their application. Furthermore, outdated documentation is available, which means that developers have to read the source code in order to interpret the documentation.

Not all messaging applications are flexible enough to allow third-party access, either through bot APIs or even a full user API. This limitation, indeed, makes it more difficult or impractical to extend current solutions but could be achieved through the use of reverse engineering. This scenario is slowly growing in the open-source community, where users create reverse-engineered solutions or APIs for free.

In an ideal messenger application, third-party applications could either be run from inside the user applications by demand, i.e. by user request, through a list of pre-accepted providers, being one them is Ethereum or by having a connection with the messaging application through an open source API that can access information regarding the users' conversation data in real time. This would allow the user to either have a running node that is controlled by the messenger app or an external independent application dedicated to creating and syncing nodes. The latter would require messaging API access to listen for transaction requests. Although some companies already provide APIs, they are mostly Bot APIs, which do not provide user peer-to-peer conversation access and are not run from user devices but from the cloud.

4.2 Ethereum viability in mobile devices

Section 2.5 discussed that Ethereum offers an extra layer of development flexibility through their own platform/protocol, which is based on Blockchain and focuses on addressing some of the Bitcoin limitations. By using this platform, users could have more visibility over certain conditions that are needed to fulfil a transaction as long as the original source code used is publicly provided, such as conditions before making a transaction or even complex conditions through contracts.

As shown in section 3.5.2, it is possible to connect cryptocurrency transactions for messaging applications in section 3.5.2 with a few extra lines of code; ideally, a similar approach is not available for traditional bank applications, as they do not allow users to initialise transactions from the messaging application itself. Currently, such integration is possible for non-trusted third party applications via some cryptocurrency platforms that provide integration for several mobile platforms, minimising the gap of integration between both applications.

4.2.1 Consumption measurement

Some mobile devices have the option to expand their storage capacity through the use of external cards, although most devices do not come with this storage allocated but have it ready for when a user decides to buy it instead. The solution application should not rely on the fact that the storage was expanded but should assume that the user does not rely on much available storage space for the application. This will help prevent users from uninstalling the application because it consumes generous storage space. To keep data as small as possible, it should periodically delete old and unused data from the Blockchain, such as old headers, and only preserve the latest information available.

Any device could serve as a node on the Ethereum Blockchain as long as it has access to the internet; it could join the network at any point in time. In mobile phones, the internal storage can vary from 12 GB to hundreds of GB. The size of a full Ethereum Blockchain node, i.e., the history of all blocks in the chain and states takes about 400 GB of storage as of April 2018 according to BitInfoCharts.

In order to reduce the storage space required, when starting a node the application will specify that the current node must run in a light mode—i.e. that this node is not going to download the entire Blockchain but only the headers for the blocks without the block and state. This will enable the node to require a smaller storage space on the device while still being up-to-date with other full-nodes and even receiving new headers.

The difference between this light mode and a full node, besides the data not being downloaded into the device storage, is the node processing. In a fully downloaded node, all blocks would be fully verified but the light mode would only download the transaction receipts and the latest history state. By running a light mode, the size of the storage used for Blockchain data on a device would drop from 400 GB to 700 MB, according to data retrieved from BitInfoCharts (data gathered in April 2018).

As a way to reduce consumption of storage on mobile devices, two other approaches could help, the first being connecting to the network through an external node provided by a third party entity. This external node, similarly to the node instantiated locally, would be used to send transactions already locally signed to the network for further evaluation from other peers. This way, no block synchronisation would need to be performed locally, considering that the external node is synchronised to the network chain. There are some providers that allow connections to nodes in the cloud, one of them being the Infura, which provides an HTTP connection to a running node connected to the network out of the box for connection. Through the usage of the external node, the storage consumption is reduced significantly, as no information gets stored on the user's device apart from the accounts stored on Keystore.

The second option is a new mechanism called sidechain, which is a separate

Blockchain attached to the main Blockchain through the use of a two-way peg, allowing assets to be interchangeable across different chains.

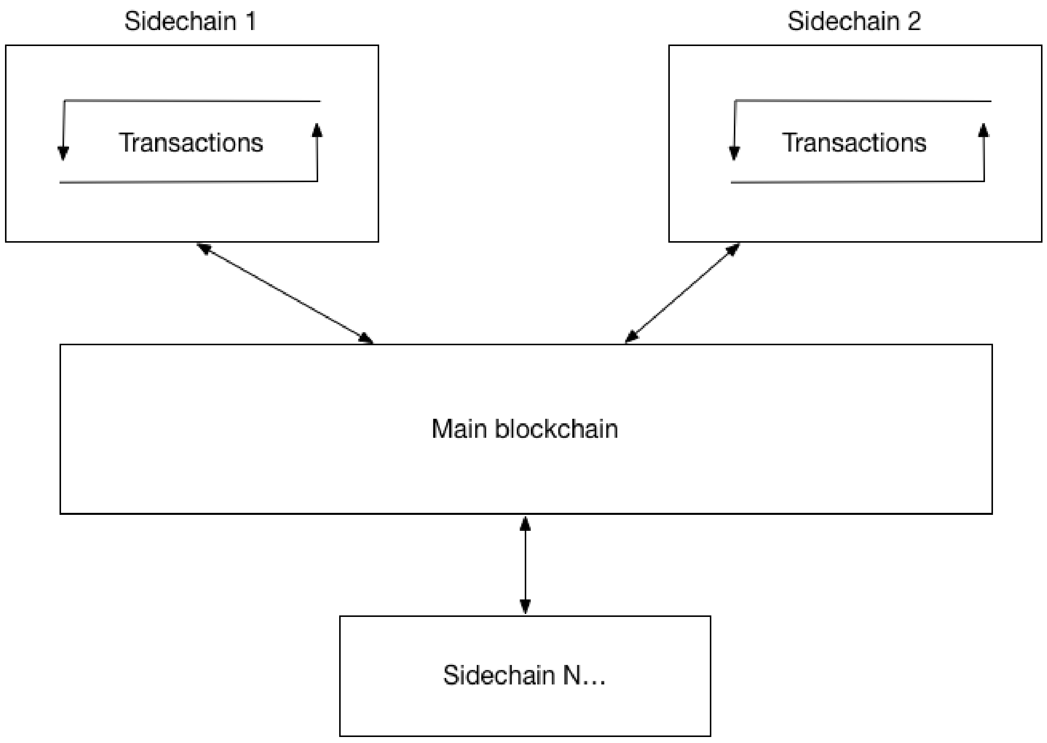

Figure 4.1: Sidechain Structure

The benefit of using sidechains comes in terms of scalability, as they were created to tackle scale issues of the main Blockchain. Finding a Proof-of-work (PoW) for a block may be expensive and slower due to the block creation time interval, as seen in section 2.4.3. These chains have their miners, meaning that they can find the next PoW in a smaller amount of time and potentially lower the processing price from that of the main chain miners, as seen in section [Back et al., 2014], and update the mainchain if required, i.e. by transferring the asset back to the main chain. However, since the sidechains are pegged to the main Blockchain, they are also independent with respect to security, meaning that, similarly to the Blockchain, they also need to have a consensus mechanism to defend against malicious attacks. One mechanism implementation option is the use of a central authority responsible for locking and releasing coins upon request from the chains—i.e. if one account sends a certain amount to an account on a sidechain, this central authority receives the value, locks the asset from the sender and releases the amount on the receiver chain. This implementation, however, raises security issues in a situation in which this central authority is the target of a malicious attack, releasing funds to a different account address. As an alternative, a Federation, or group of servers, acts as a chain mediator, validating or releasing funds if the majority of the peers in the group decide that a particular block is valid.

Although the mechanism of sidechains may help with storage consumption issues, it requires an extra layer of knowledge for users who would need to trust an external party to manage their funds on their behalf.

Chapter 5

Conclusion

As could be seen in chapter 2.1, smartphones are present in the lives of most people and communicating with others through these devices is one of the most frequently used features that their owners utilise and spend their time on. With the initial creation and propagation of SMS, people could communicate more easily through text messages with anyone around the world in a matter of seconds, using up to 160 characters per message sent.

Not surprisingly, this technology led to the creation of an 'upgraded' and more robust messaging technology, the Multimedia Messaging Service (MMS), which allowed users not only to send longer texts in terms of characters but also multimedia content, such as audio, videos, photos and more, through a message. This new technology soon became widely used; however, it was not free and users still needed to pay for each message they sent and, sometimes, more for MMS. This problem caused a boost of MIMs, commonly used and accessible to many people around the world, which, together with the constant updates that smartphones' operational systems received, became faster and better during each interaction, resulting in their widespread use among users.

This race led companies and passionate developers to always reach further, for the next step—one of which is making the possibility of conducting money transactions through messaging applications a reality, offering something that not many other applications are offering. This could be integrated using real currencies through an open API for banking applications; however, due to the lack of security of non-trusted parties, banks can hardly, if ever, be expected to abandon the security facade of their main system. In messaging applications, users do not have the option for in-app financial transaction services, which causes a delay in transactions or even prevents these transactions from being made without relying on external online services. This problem, however, can be solved through the use of a virtual currency, such as Bitcoin or Ethereum, which now has, for example on Ethereum, almost 70 million accounts.

5.1 Opinions

5.1.1 Ethereum on messengers

As shown in this research, the flexibility behind cryptocurrency technology allows companies and entrepreneurs to create and share their ideas with a broad online community. With a few lines of code, a device can be configured and able to connect to any network or to multiple networks at the same time. Users could then decide what network should be used for the new transaction and switch between them in a few interactions through indicative buttons.

The possibilities that Ethereum allows to be developed on top of their platform are indeed robust. Chapter 3 described how it is possible to integrate both solutions into one robust application. Ethereum allows what some users value the most in messenger applications: flexibility and speed when it comes to results. A simple matter of a few milliseconds could potentially negatively affect the user experience with the application; however, by separating the messenger application from the Ethereum application, the user has in his hands the best of both systems, thus making cryptocurrency transactions accessible through only a few interactions.

Apart from allowing simple account transactions, as seen in chapter 3, the usage scenario for MIM applications could be expanded to encompass e-commerce payments through the use of bots or money raising, such as charity or ICO, through a similar approach as the one that WhatsApp is currently taking with their e-commerce based account. Not far removed, channels or broadcasters, one-way information sources, can also use this as a way to raise funds for a certain product or idea. Users could, for example, check the catalogues of companies from their official bots, which would then process the incoming transaction and store the current balance in an Ethereum contract.

If the company releases the official source code of the contract, this then allows for greater transparency about the destination of the money that is sent because the code of the compiled contract is publicly available to the entire network and can be compiled locally, allowing users to compare the integrity of the published verwsion against the local compiled version. This transparency would reduce the users' fear of donating to or buying goods from untrusted sources while companies or charity entities could officially release their bots with a trust certificate, in conjunction with the Messenger Platform, in this case Telegram, to highlight a real account—as is done on Twitter, Facebook and Instagram, where the platform displays whether a certain account is a genuine account or not.

Another good scenario for Ethereum would be in groups; for each group, for instance, a new Ethereum contract could be issued that would be linked to a Telegram group. Since Ethereum requires an administrator account to create and compile a contract, the administrator or creator of the respective group account would be linked to the contract, allowing, for example, money raising, group activities or scheduled events that could only be initialised or withdrawn after a certain goal was achieved, such as at least 60% contribution from participants and the money would go to a specific account.

5.2 Security of Ethereum on mobile device

Securing data on mobile devices is always a challenge because they can be rooted any time, meaning that their default secured folders could be visible for reading purposes by any application running on the device quite easily. Users would need to take an extra measure when it comes to securing their accounts, as losing or sharing their credentials could mean losing all assets that the account holds. Although the private key is secured by a passphrase used to encrypt it, if the user uses a non secure combination, with time it could be possible to crack the credential by using brutal force, which is done by testing different combinations of passwords in order to decrypt an encrypted file. Users should always be clear about the importance of a good passphrase, which, depending on how long and complex they are, could take days or even years to crack. It is important to remember that passphrases should be easy enough for the owner to remember but not for a computer to crack and that their complexity grows exponentially as new words are added to the combination.

Furthermore, keeping a backup of the keys is also important, in case the keys stored on the user's device are wiped (or the entire device itself) or are corrupted by an external factor, such as a bug or a virus. Ethereum and Bitcoin always recommend that their users make an offline copy of their keys so that they could be accessed even if the main gate to the network is corrupted, which would be the mobile application itself in this case.

5.3 Future possibilities for messaging applications with cryptocurrency integration

Messaging applications are quickly evolving to satisfy a particular user need. They have grown from an application that handled text messages to a more robust one that is capable of handling multimedia messages, files and, recently, location sharing in chats. Online payments are a need that some users have. In an ideal scenario, traditional banks would provide a public interface capable of making transactions from non-trusted third-parties, however, due to security reasons this cannot be achieved. Cryptocurrency, on the other hand, since it was publicly presented with details about how it should work—using proof-of-work to validate transactions—removed this gap between cash systems, allowing non-trusted third-party solutions to not only access transaction information but make new cryptocurrency transactions.

This flexibility naturally leads to the adoption and utilisation of cryptocurrency payment systems and has been widely used among users (see section 2.3.3). Unique solutions have emerged inspired by this system, either by integrating it entirely or by extending its model onto something else, such as the creation of new coins.

The growth of interest in exchanging coins by users through mobile devices can be broadly explored in an application that has live integration with the network—and this integration is completely possible to achieve, as can be seen in chapter 3. Sending cryptocurrency through chat applications could be done comfortably from a user perspective as much so as sending a photo. This solution could be implemented and distributed through application stores, targeting both experienced cryptocurrency users and users that have no previous knowledge about this new technology.

Bibliography

[Back et al., 2014] Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J., and Wuille, P. (2014). Enabling blockchain innovations with pegged sidechains.

[Balasubramanian et al., 2018] Balasubramanian, B., Baxley, D., Birch, J., Bondar, B., Botes, C., Cole, S., Crowe, M., Cullimore, B., Doloc, M., Gluck, M., Guillen, A., Hagen, A., Jackson, N., Janfaza, F., Hartman, S., Konanur, S., Kuhn, E., Lopez, C., Martiesian, J., McArthur, S., McGill, D., Molitor, D., Moore, J., Mott, S., Pitterle, B., Reef, R., Rice, G., Smith, T., Syed, M., Toth, L., Townsend, L., Urtan, A., Blarcum, C. V., and Whittemore, A. (2018). Mobile and digital wallets: United states landscape and strategic considerations for merchants and financial institutions. Technical report, U.S. Payments Forum.

[BBC, 2002] BBC (2002). Happy birthday text!

[Blockchain, 2018] Blockchain (2018). Blockchain wallet users. Retrieved from10th Dec 2018.

[Burke, 2016] Burke, K. (2016). How many texts do people send every day (2018)? Retrieved from 10th Oct 2018.

[Cook, 2018] Cook, J. (2018). United kingdom fintech startup revolut reaches 2 million users.

[Das, 2016] Das, S. (2016). India set to descend into cash chaos. Retrieved from 14th Sept 2017.

[Data, 2018] Data, F. (2018). Retrieved from 10th Oct 2018.

[Deloitte, 2017] Deloitte (2017). Global mobile consumer survey 2017: United kingdom cut. Retrieved from 8th Oct 2018.

[Ethereum Charts, 2017] Ethereum Charts (2017). Ethereum / ether (eth) statistics - price, blocks count, difficulty, hashrate, value. Retrieved from 13th Sept 2017.

[Facebook, 2015] Facebook (2015). Send money to friends in messenger.

[Facebook, 2018] Facebook (2018). Why messaging businesses is the new normal.

[Felföldi, 2017] Felföldi, Z. (2017). Introduction of the light client for dapp developers. Retrieved from 9th May 2018.

[Ferguson, 2008] Ferguson, N. (2008). *The ascent of money.* The Penguin Press.

[Garret, 2017] Garret, O. (2017). India is likely to become the first digital, cashless society. Retrieved from 12th Sept 2017.

[GSM History, 2018] GSM History (2018). Gsm history. Retrieved from 8th Oct 2018.

[Hileman and Rauchs, 2017] Hileman, G. and Rauchs, M. (2017). Global cryptocurrency benchmarking study. Technical report, Cambridge Centre for Alternative Finance.

[Incrybit, 2018] Incrybit (2018). Crypto exchange evaluation.

[Investing, 2018] Investing (2018). All cryptocurrencies.

[Jowitt, 2017] Jowitt, T. (2017). Tales in tech history: The sms text message at 25. Retrieved from 8th Oct 2018.

[Kafka, 2016] Kafka, P. (2016). The app boom is over.

[Leung, 2016] Leung, O. (2016). Chapter 1: What is mobile instant messaging? Retrieved from 8th Oct 2018.

[LINE, 2018] LINE (2018).

[Line, 2018] Line (2018). Line token economy.

[Livingston, 2016] Livingston, T. (2016). talk title. TechCrunch Disrupt NY.

[Mauldin, 2017] Mauldin, J. (2017). Raoul pal, paying attention. Retrieved from 14th Sept 2017.

[Mclean and Deane-Johns, 2016] Mclean, S. and Deane-Johns, S. (2016). Demystifying blockchain and distributed ledger technology – hype or hero? *Computer Law Review International*, 17(4).

[MIM Reach, 2015] MIM Reach (2015). Mobile messaging reach 1.4 billion worldwide in 2015. Retrieved from 8th Oct 2018.

[Nakamoto, 2009] Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. Technical report, Bitcoin.org.

[Pasquet and Gerbaix, 2017] Pasquet, M. and Gerbaix, S. (2017). Instant payment versus smartphone payment. In *2017 Third International Conference On Mobile And Secure Services (MobiSecServ)*, page 2.

[Reiff, 2017] Reiff, N. (2017). What is erc-20 and what does it mean for ethereum? Retrieved from 13th Sept 2017.

[Siegel, 2017] Siegel, D. (2017). Understanding the dao attack. Retrieved from 13th Sept 2017.

[Statista, 2013] Statista (2013). Average number of mms messages sent per user per month in the united states from june 2005 to december 2011 (in billions).

[Statista, 2018a] Statista (2018a). Most popular mobile messaging apps worldwide as of july 2018, based on number of monthly active users (in millions). Retrieved from 10th Oct 2018.

[Statista, 2018b] Statista (2018b). Number of blockchain wallet users worldwide from 1st quarter 2015 to 3rd quarter 2018.

[Statista, 2018c] Statista (2018c). Number of smarphone users worldwide from 2014 to 2020 (in billions).

[Tang and Hew, 2017] Tang, Y. and Hew, K. F. (2017). Is mobile instant messaging (mim) useful in education? examining its technological, pedagogical, and social affordances. *Educational Research Review*. Retrieved from 8th Oct 2018.

[Telegram, 2018] Telegram (2018). Android client source code.

[UlsterBank, 2018] UlsterBank (2018). Mobile app security. Retrieved from 12th Oct 2018.

[Vyas, 2012] Vyas, S. (2012). Impact of e-banking on traditional banking services.

[Wilmoth, 2017] Wilmoth, J. (2017). Gpu mining 'important driver' for future amd/nvidia growth: Market analysts. Retrieved from 3rd May 2018.

[Worldwide SMS Market, 2014] Worldwide SMS Market (2014). Worldwide sms market. Retrieved from 8th Oct 2018.