

UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Computer Science Curriculum

Sevil Guler

Secure Bitcoin Wallet

Master Thesis (30 ECTS)

Supervisor: Sead Muftic, Prof

Supervisor: Vitaly Skachek, Dr

Acknowledgements This research work has been performed in the Department of Communication Systems, Royal Institute of Technology, Stockholm, Sweden. This thesis work spans five months of extensive research effort with the encouragement and appreciation of numerous people. I wish to express my deep gratitude and thanks to all those people in this acknowledgements.

I would like to express my very great appreciation to Prof. Sead Muftic for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated. Without his assistance and encouragement, the thesis work could not be an easy task for me. Thanks to Prof. I am profoundly thankful to Dr. Vitaly Skachek for agreeing to be my examiner and his generous help and understanding. His support throughout the project was outstanding and unconditional. I would like to also thank to Ms. Nazri Abdullah, Dr. Ghafoor Abbasi and thanks to all SecLab colleagues.

Lastly, I pay my sincere regard to all my friends who supported me in any aspect during my stay in Sweden.

Secure Bitcoin Wallet

Abstract:

Virtual currencies and mobile banking are technology advancements that are receiving increased attention in the global community because of their accessibility, convenience and speed. However, this popularity comes with growing security concerns, like increasing frequency of identity theft, leading to bigger problems which put user anonymity at risk. One possible solution for these problems is using cryptography to enhance security of Bitcoin or other decentralised digital currency systems and to decrease frequency of attacks on either communication channels or system storage. This report outlines various methods and solutions targeting these issues and aims to understand their effectiveness. It also describes Secure Bitcoin Wallet, standard Bitcoin transactions client, enhanced with various security features and services.

Keywords:

Android, Security, Bitcoin, Mobile Payment, Java, Instant Messaging, Wallet, BTC, Virtual Currency, Cryptography, Push Notification, PKI, Shared Preference, SQLite, Security Architecture, Software Development, Mobile Development, Public Ledger, Miner

Kokkuvõte:

Virtuaalvaluutad ja mobiilne pangandus on tehnoloogilised uuendused, mis on rahvusvahelises kogukonnas saamas kasvavat tähelepanu oma kättesaadavuse, mugavuse ja kiiruse tõttu. Populaarsuse kasv on kahjuks kaasa toonud ka suurenenud turvariski identiteedivarguste näol, tekitades ohu kasutajate anonüümsusele. Riske on võimalik vältida, kasutades krüptograafilisi meetmeid Bitcoin ja teiste hajutatud digitaalsete valuutade vastaste rünnete vähendamiseks sideliinil ning hoiustamisel. See ülevaade koondab erinevad meetodid ja lahendused selliste rünnete vastu ning uurib nende tõhusust. Lisaks kirjeldatakse turvalist Bitcoin rahakotti (Secure Bitcoin Wallet), mis on standardne Bitcoin ülekannete klient koos tõhustatud turvaomaduste ja -teenustega.

Märksõnad:

Android, turvalisus, Bitcoin, mobiilne pangandus, Java, välksuhtlus, rahakott, BTC, virtuaalvaluuta, krüptograafia, tõukesõnumid, avaliku võtme taristu, ühiseadistused, SQLite, turvaarhitektuur, tarkvaraarendus, mobiilne arendus, avalik pearaamat, kaevur

Contents

1	Introduction	7
1.1	Problem Statement	7
1.2	Goals and Achieved Results	7
1.3	Research Methodology	8
1.4	Audience	9
1.5	Limitation	9
1.6	The Structure of The Thesis	9
2	Background	10
2.1	Android Mobile OS	10
2.2	SSL Protocol	10
2.3	Openfire Server	11
2.3.1	XMPP and Smack APIs	11
2.3.2	User Service Plugin	12
2.4	Bitcoin System	13
2.5	How Bitcoin System Works [16, 12]	13
2.5.1	Components	14
2.5.2	Bitcoin Wallet	15
2.6	Push Notifications	17
2.7	PKI for Mobile Environments	17
2.7.1	Symmetric Key Cryptography	17
2.7.2	Asymmetric Key Cryptography	18
2.7.3	X.509 Certificates	18
2.7.4	PKI	19
2.7.5	PKI in Current Mobile Systems	19
3	Security Architecture for Bitcoin Transactions	20
3.1	System Components	20
3.1.1	Crypto Service Provider (CSP) [18, 20, 21]	21
3.1.2	Client Application	21
3.1.3	User	21
3.1.4	Instant Messaging (IM) Server	21
3.1.5	Payment Server	22
3.1.6	Wallet Server	22
3.1.7	Bitcoin Network	22
4	Sytem Protocols and Operations	23
4.1	Local and Remote Registration Protocol	23
4.2	Local and Remote Authentication Protocol	25
4.3	The Certificates Protocol [22]	26
4.4	Notification Messages Protocol	27
4.5	Instant Messaging Protocol	28
4.6	Listing Balance and Transactions Protocol	30
4.7	Requesting Payments Protocol	31
4.8	Making Payments with Requests Protocol	33
4.9	Making Payments (without Request) Protocol	34

5	Design of the Demo	36
5.1	Launching	36
5.1.1	Registration	37
5.1.2	Registration Errors	38
5.1.3	Login	39
5.2	Main Page/Main Menu	40
5.3	Secure IM System	41
5.4	Secure Payments	42
6	Evaluation of the System using Engineering Standards	44
6.1	Scope	44
6.2	Economy	44
6.3	Reliability	44
6.4	Usability	44
6.5	Performance	44
6.6	Objectives and Success Criteria of the Project	44
6.7	Ethical	44
6.8	Security	45
6.9	Social and Political	45
7	Conclusion	46

List of Figures

1	Overview of SSL Handshake Process	11
2	System Components	20
3	Components and Protocol for User Registration	23
4	Components and Protocol for Local and Remote Authentication	25
5	Protocol to Request/Receive Certificate	26
6	Notification Messages Protocol	27
7	Instant Messaging Protocol	28
8	Components and Protocol to List Balance and Transactions	30
9	Component and Protocol for Requesting Payments	31
10	Componential Protocol for Making Payments(with Request)	33
11	Components and Protocol for Making Payments(without Request)	34
12	Registration Panel	37
13	Registration Error 1	38
14	Registration Error 2	38
15	Login Panel	39
16	ID Card	40
17	Main Menu	40
18	Fetch Message Panel	41
19	Send Message Panel	41
20	Secure Payments Panel	42
21	Request Payments Panel	42
22	Send Payments Panel	43
23	Balance and Transactions Panel	43

1 Introduction

Virtual currencies have been defined in 2009 by an anonymous person named Satoshi Nakamoto as a decentralized digital peer to peer payment system. Bitcoins, in particular, have increasing attention and two observations are attributed to its growing notice. First, virtual currency has great potential to become the most widely used means for transactions in the future. Second, it might become a tool used by criminals to transfer and store illegal funds without any control from law enforcement agencies and other authorities.

Instead of relying on central authorities virtual currency is based on the idea of using cryptography to control its creation and transactions. It is implemented by a long string of numbers and letters sent over the Internet without using a bank. Virtual currencies offer a different approach from what governments do and the banking institutions controlling the world play with rates, organized fraud, and bailouts that do nothing for the ordinary person. Hence, virtual currencies can be used instead of a government-issued currency to purchase goods and services in the real economy.

Virtual currencies also have a possibility to transfer money anywhere in a very easy way and they allow users to be in control of their money. However, such great features of flexibility and usability also come with increased security concerns.

Furthermore, nowadays with increasing popularity of mobile phones for communication, most people are equipped with mobile devices. While mobile phones were earlier used mainly for telephone services, today they are used for other services including delivery of information. With increased usage of such improved services, mobile phones have extended their scope making users' life easier, like online-banking, mobile payments, etc. Hence, mobile phones provide the best way for using virtual currencies in payment systems.

1.1 Problem Statement

Bitcoin is an emerging crypto-currency system with a potential to become a payment system for modern day businesses. Currently, Bitcoins can be traded on Exchange Services for traditional currencies, such as EUR, USD, GBP, etc. With the trending growth of Bitcoins, there exists a strong requirement for combining Bitcoin wallets with mobile payment gateways. However, there is a big gap bridging these two insecure systems.

Current Bitcoin System lacks many attributes of a useful currency. First of all, in current Bitcoin System transactions take at least 10 minutes to be completed which makes usage of Bitcoin as a money in real life infeasible. For example, if user wants to use Bitcoins in any store she/he can't leave the store until the payment completed. Furthermore, in current Bitcoin system there is no authorization. Consider the problem of theft. Once stolen cash enters the system, little can be done to reclaim it.

1.2 Goals and Achieved Results

This thesis addresses security concerns when coupling Bitcoin wallet with the mobile payment gateway by building an Android application that makes secure combination of virtual currencies and mobile payments. The application targets lack of the Bitcoin System and provides efficient solution to them. The application also provides secure communication between mobile phones based on cryptographic protocols by using strong

encryption schemes. The work also aims to provide user-friendly interface to facilitate easy usage of the new application implementation in the context. Android is the chosen mobile operating system for two obvious reasons: to target more users and to increase viability of testing of the implemented system in real time.

The application is called '*Secure Bitcoin Wallet*'. It is designed to be simple to use, but it is based on complicated background security architecture. It provides higher level of security to users for virtual currency transactions. The application is ideal for situations where anonymity and security is a major concern for personal and business purposes. It aims to protect users from commercial espionage, governments, mobile phone companies, or any other third parties who want to collect sensitive information or private correspondence related to financial transactions.

As part of the implementation, an Application Programming Interface (API) called "*Mobile Crypto Services Provider*" was also built which provides cryptographic services for secure communications between mobile phone and its storage unit. To augment the primary requirement of providing secure Bitcoin mobile wallet functionality, a *Secure Instant Messaging* application is also included using the underlying Mobile Crypto Services Provider with the newly built Android application.

1.3 Research Methodology

In this research, an outcome based methodology called design science is used. Design science involves application of various approaches, methods and techniques leading to development of an innovative idea, knowledge or product. This is to be achieved through design of novel artifacts and thereafter using and evaluating the artifacts to better understand the behavior of the system.

The process begins by defining the problem and branches into design suggestion, development, evaluation and conclusion. In our project, the most evident security augmentation service that can be a solution for Android devices is the storage of secure credentials in internal memory. However, this solution leads to several other security issues. According to Android documentation every application has its own internal storage area and only the owner of this area can use this storage. Though, even the owner of the phone cannot access data, meaning data can be saved securely, there are many security vulnerabilities and violation of Android security. Therefore, we have chosen internal memory of the mobile phone with security enhancement by using cryptography. The motivation behind this is that even when the mobile device is lost or stolen or an eavesdropper entraps the traffic during transit from mobile device to the application service provider, credentials will not be readable by others.

Furthermore, since in our system we have many parties that need to communicate with each other, communication channels based on Crypto Service Provider are used for security enhancement of the system. Everything, be it on air or on the mobile storage, is encrypted.

The research work is done after understanding the previous work in this field, industry standards, and by broad exploration of KTH and Tartu University library, IEEE publications, articles and journal papers, online websites, Google scholar, search engines and web directories.

1.4 Audience

Owing to the increased competition in the mobile application development market, a number of enterprise level financial sector applications lack comprehensive security solutions in their deployed systems. This research work targets all those bodies (i.e. financial institutions and organizations) who want to add Bitcoin option in their applications with secure solutions that mitigate threats to their systems with ease and mobility features. We expect that the proposed design and development results of this research play an important contribution to this dynamic field and pave way to build secure solutions for mobile payment applications. The chosen approach is scalable; can interlink with other platforms and modular in the sense that it can easily be deployed or added on to any application that require security services. We also expect the security solution will be transparent to users and it will provide end-to-end application layer security.

1.5 Limitation

Like any time-bound, result-specific research work, this work also has few inherent limitations. This work is primarily concerned with security services for mobile applications based on Android platform. It aims to protect applications and the users' authenticity, confidentiality and integrity by utilizing cryptography. Due to time constraints, the scope of this work was limited to applications developed for Android platform. Since wireless data is passing through the air interface faces almost the same security threats as wired data, security based development is the main concern in this research. However, the limited wireless bandwidth, battery, computational power and memory of wireless devices introduce further limitations to the implementation of security mechanisms.

1.6 The Structure of The Thesis

Chapter 2 provides a brief overview of the Bitcoin and mobile technologies emphasizing the underlying components, such as SSL, Openfire server, Public Key Infrastructure and Bitcoin wallets. It provides background information needed for understanding of the following sections.

Chapter 3 provides an insight into the security architecture of the APIs that have been built for Bitcoin transactions, along with a description of every component that has been built. Chapter 4 articulates different functionality of the Bitcoin Wallet based on the APIs outlined in Chapter 3. Additionally, Chapter 5 describes various use-cases of the system. Chapters 3,4 and 5 together highlight author's contributions in terms of system design and software implementation.

Chapter 6 provides an overview of design aspects (from a user interface point of view) of the Bitcoin Wallet and the Secure Instant Messaging application, both using crypto APIs. Chapter 7 contains evaluation of all three software modules (API, Bitcoin Wallet and IM) built as a component of the applications in terms of software engineering standards.

2 Background

In this chapter background concepts related to the Secure Bitcoin Wallet System are discussed. In order to provide a robust and efficient system, we use all technologies described in the subsequent sections of this chapter.

2.1 Android Mobile OS

Android is one of the most popular mobile platforms. It is Linux-based and represents multi-processing and multi-threading operating system. Android has sand-boxed applications environment, which provides tightly controlled set of resources for programs to run in, such as scratch space on disk and memory. Network access, the ability to inspect host operating system or read from input devices are usually prohibited or heavily restricted. The aim of sand-boxing is to provide security by isolating an application to prevent dangerous parties like malware, intruders, etc. from interacting with the protected application. All applications run separately without any interactions between them what poses a problem in case when developers want applications to interact with each other.

Android generally uses permissions for applications security management. However, this concept is being questioned by researchers, since security flaws are cropping up currently. For example, the first SMS Trojan detected for smartphones running Android was detected in 2010 which was a mobile application displaying pornographic content to users. Once the user installs the said application, script files were generated and enabled the application to modify its configuration settings. The application was also able to randomly send messages, imposing charges to the user without their explicit validation, thus violating Android security system permissions. Similarly, security flaws were explained by other researchers also in violation of restriction permissions of Android.

There are typically no solutions to such problems, since these are caused by rudimentary structure of the Android operating system. Though there are many anti-virus applications created for Android, they are not efficient enough in curtailing security violations of Android systems.

2.2 SSL Protocol

SSL is a de-facto standard security platform for secure end-to-end communications which provides confidentiality, integrity and authenticity. In today's Internet, X.509[2] is a standard for public key infrastructure (PKI), where user has a preloaded list of root certificates (top level PKI CAs) and any certificate issued by those authorities is accepted by the user. In modern browsers it is implemented by having certificates of root CAs(Certificate Authorities) embedded in the web browser. Some browsers, such as Google Chrome, also use certificate pinning [3] for the same purpose.

When a software attempts to connect to a SSL secured server, it starts with a handshake phase between the client and the server, which is called "SSL handshake"[5]. The purpose of this process is to establish a symmetric session key between the client and the server, which is consequently used to encrypt all of the transmitted data. The overview of the SSL handshake process is shown in Figure 1.

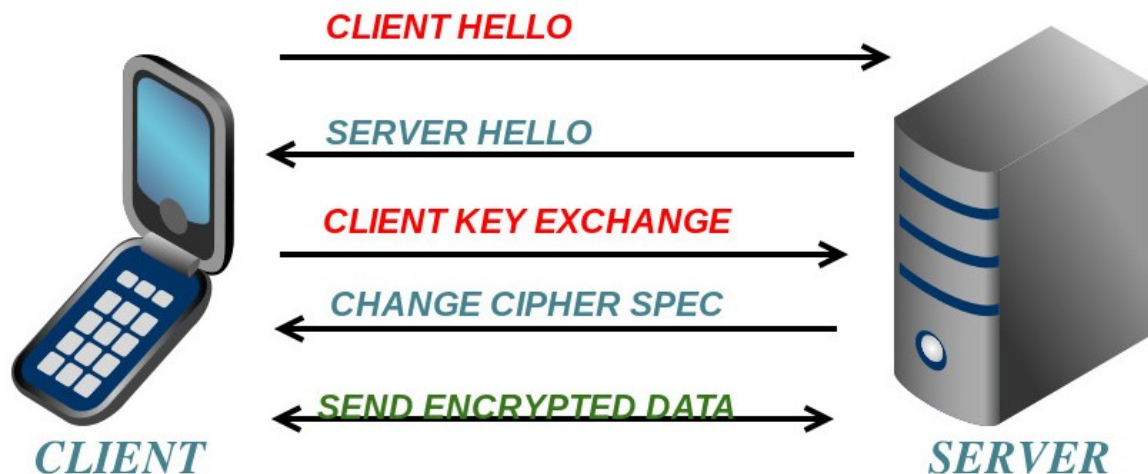


Figure 1: Overview of SSL Handshake Process

In the SSL handshake process the client first sends a hello request message to the server, which includes some security parameters of the client. The server then processes the client's hello message and responds with a server hello message. The server's hello message also includes some security parameters which depend on the client security parameters, as well as its public-key certificate. The client then validates the certificate of the server. If it trusts the certificate, it will create and send back a symmetric session key to the server using the server's public key. Consequently, the server uses its private key to decrypt the symmetric session key and send back an acknowledgement encrypted with the session key. Finally, the server and the client can exchange data securely with the session key.

2.3 Openfire Server

Openfire is a real time collaboration (RTC) server licensed under the Open Source Apache License. It is an instant messaging and group chat server that uses XMPP protocol written in Java [6, 7]. In general, Openfire server has its own administration application to control user's registration and chatting. It doesn't provide any functionality for the developer to create their own registration module from their application. However, it has a plugin interface that provides an alternative approach for the administrators to extend the server by uploading any available Openfire plugin.

2.3.1 XMPP and Smack APIs

Extensible Messaging and Presence Protocol (XMPP) is a communication protocol for message-oriented middleware based on XML (Extensible Markup Language)[9, 10]. Usage of it might be a big challenge, because of complexity of XML parsing. However, there are some good API's, like Smack API, which makes developers' work easier. Smack API is an Open Source XMPP (Jabber) client library for instant messaging [8]. It is a

native Java library, which can be embedded into applications to create anything from a full XMPP client to XMPP Server integration, such as sending notification messages and presence-enabling devices. After version 4.1, Smack API does not work on Android devices. Therefore, Smack API was repacked for Android devices called aSmack.

2.3.2 User Service Plugin

User Service Plugin provides the ability to manage users by sending an HTTP request to the IM server[11]. It is intended to be used by applications which automate user administration process. This plugin is used by applications that administers user without using Openfire Admin console. In order to provide a standard way for accessing data, REST protocol is used. All REST Endpoints are secured by Basic HTTP Authentication or by using a shared secret key.

Basic HTTP Authentication : Basic HTTP Authentication protocol requires providing user-name and password for the Openfire Administrator's account in a header request in order to access the endpoints.

Shared Secret Key : Shared Secret Key technique requires sending secret key in the header request to access the endpoints.

Vulnerabilities : Both techniques are not secure. Plugin uses HTTP Connection which is not encrypted. Data on the communication channel is not encrypted and hence confidentiality is not provided. Security of the entire instant messaging system relies on the username and password of the Administrator, what make security very weak. An attacker, if in possession of Administrator privileges which include full access to the Server, can delete or update existing users by using their user name and email addresses and also can add any number of users to the system.

2.4 Bitcoin System

Bitcoin is complex topic encompassing cryptography, software engineering and economics and is believed to provide safe transactions and avoid double-spending through the clever use of public-key cryptography. Bitcoin is the first decentralized virtual currency with significant increase in usage and popularity. It was introduced in 2009 by Satoshi Nakamoto and it received a lot of attention from the media and from many considerably big companies, such as WordPress, Paypal, Ebay. These companies are the first to accept Bitcoins as a medium for currency payments. The number of companies realising the benefits of Bitcoin and accepting it as a payment currency is increasing day by day and at the time of writing these report 14.179.200 Bitcoins are in circulation [15] and each Bitcoin is worth \$ 234.10 USD [14].

In the Bitcoin system transactions are distributed to the whole network without encryption and they are available to all nodes in the peer-to-peer network. This leads to privacy concerns. Every transaction is publicly logged, i.e. anyone can see the transactions between two users. Although addresses are random numbers, network analysis, surveillance, or just searching the addresses over the Internet can disclose the actual identity of clients. This problem of anonymity in Bitcoin might have a simple solution. Using a new address for every transaction is one of the officially encouraged methods to make these attacks more difficult. Furthermore, TOR(The Onion Router), whose development is in its inception, is one of the recommended solutions for this problem.

All in all, even though there are several benefits of using Bitcoins, just like any other crypto-currency, it has been associated with lots of scams, hacks/thefts, defunct "stock exchanges," and reported loss of wallets containing massive amounts. Although Bitcoin currencies have many security issues as indicated before, it is still gaining popularity. Therefore, further research has to be undertaken to make it better with enhanced security.

To understand the concept of Bitcoins we define how it works and examine the background of the system in the following section.

2.5 How Bitcoin System Works [16, 12]

Suppose user X has user Y's account number. What prevents user X to access and use money in Y's account? In traditional banking services a pen and paper check ensures this. A signature must be verified to prove that the sender's signature is real. With Bitcoins, this is ensured by using cryptographic keys rather than handwriting. Bitcoin system uses Public Key Cryptography to solve this problem and whenever a new account number is created, a private key will be mathematically associated with this account number. Transaction then can be validated by Bitcoin community by using public key of the signature's owner. The mathematical signatures prove the identity of the transacting user and also prevent non-repudiation if the sender tries to deny making the transaction in the future. In traditional banking system, Bob writes two checks each amounting to total balance in his bank account. In real time cases, the bank pays the first person attempting to cash the check, but refuses the second because Bob's account will be empty. In our context the question is who will get paid in Bitcoin System? Unfortunately, tracking orders is very hard in the Bitcoin system because network delays may cause transactions to be requested in different times at different places what effectively allows Alice to spend money twice.

2.5.1 Components

Transactions : All transactions can be performed by signing a hash of the previous transaction and the public key is received and is to be validated by the receiver. Receiver verifies the signatures to validate the chain of ownership. The drawback of this is the inability of the receiver to identify double spending of coins. To ensure this, receiver should verify that the sender did not sign any earlier transactions and should be aware of all transactions from all possible sources and decide which transaction has arrived first. To prevent double spending, all transactions are publicly published and timestamp server (it is not in our scope) is used for participants to agree on a single history of the order they have received. Majority of the nodes should be in agreement on which transaction have arrived first and then only that transaction will proceed.

Public Ledger : Bitcoin system eliminates centralized control, so every participant must maintain their own copy of the ledger. This results in the scenario where everyone can see everyone else's balances. The system uses account numbers called Bitcoin address and not names; so there is some level of anonymity maintained. Since everyone maintains their own ledger a mechanism should be in place to make sure that all ledgers are kept in sync when the money is transferred. For this, when money is being transacted a message is broadcasted with the user account number to all parties. Once the message is received the entire world updates the ledger. The Bitcoin system uses a peer-to-peer payment network and hence nodes can leave and later rejoin the network. Upon reconnection, a node will download and verify new blocks from other nodes to complete its local copy of the block chain.

Miner : Mining is the process of adding transaction records to Bitcoin's public ledger of past transactions called the blockchain. Individual blocks must contain a proof of work to be considered valid. Proof-of-work cryptographic computational problems are solved by these entities and miners are people who use software or hardware to achieve this. This proof-of-work is verified by other Bitcoin nodes each time they receive a block. Every such new block created by the miner is verified by the one who creates it. Each of those blocks contains a reference to the previous block; thus they form a blockchain. By extending the blockchain, the miner attests that he has accepted all previous blocks in the chain. Miners also get rewarded in Bitcoins for solving the proof-of-work cryptographic problems. The blockchain serves to confirm transactions to the rest of the network as having taken place. Bitcoin nodes use the blockchain to distinguish legitimate Bitcoin transactions from attempts to re-spend coins that have already been spent elsewhere.

Blocks and BlockChain : In a Bitcoin system, block constitutes the list of all executed transactions and the complete record of transactions in the shared public ledger obtained by appending all the blocks forms the blockchain. This structural organization aids in removing the necessity of a centralized trusted authority. All transactions recorded by the blocks have to be approved by the Bitcoin network once it is added to transaction pool. Every block includes the hash value of the previous block and thus link the current block with the previous one. Recent transactions and a random number known as nonce are used to produce the next block that is used for proof-of-work system.

TimeStamp Server : TimeStamp Server is used for proving that data existed at a particular time. It takes hash of a block of items to be timestamped and publish the hash. Each timestamp also includes the previous timestamp in its hash.

Proof-of-Work : The Bitcoins network has to prevent miners from hashing unauthorized transaction blocks or all the Bitcoins would be mined in minutes. The Bitcoin protocol makes this possible by using Proof-of-work. The proof-of-work system is used to implement timestamp server based on cryptographic hashing. System relies on double-hashing fixed size block header by using SHA-256 algorithm and specific target number which needs to be bigger than 256 bits. Only such hashes will be accepted by the system. The miner has to append a random number to the new block and make sure the hash value of the new block begins with a series of zeros which can not be created by brute-force. During the process of block chaining, verification is done using previous user's public key and the miners use their private key for signing the block which they create or mine.

2.5.2 Bitcoin Wallet

Bitcoin Wallet is a digital wallet that uses digital currency as a value. It enables users to have their Bitcoins always in their possession. Bitcoin Wallet is a software that provides services to users making it possible to transact worldwide payments for free. As in real-life, Bitcoin Wallet must also be secured and not be used by unauthorized people. There are many aspects that should be taken into consideration:

Access Management : Users' Bitcoin Wallet should be protected and should not be accessible without authorization. In general, the most common and popular method for authentication is using passwords. Security vulnerabilities that are the results of users' wrong and weak choices of passwords are not in this thesis's scope. Though this is not the best solution for providing the highest level of security, it is the cheapest and easiest way for providing security; assuming that the users will be careful enough to choose passwords which are difficult to guess or hacked.

Management and Protection of User Security Profiles : [17] User profile should be stored at the server and at the same time either on device storage system or in the network. In general, Android system offers five types of storage options:

Shared Preferences :

Store private primitive data in key-value pairs.

Internal Storages :

Store private data on the device memory.

External Storage :

Store public data on the shared external storage.

SQLite Databases :

Store structured data in a private database.

Network Connection :

Store data on the web with your own network server.

Saving and retrieving data from these security profiles should be carefully considered. In Bitcoin wallet system, data must be saved securely. External storages are world-readable and can be modified by the user by using USB mass storage to transfer files on to a computer. Network storage can be used to store and retrieve data on application's own web-based services. Hence, Shared Preferences options can be used to save and retrieve persistent key-value pairs of primitive data types such as booleans, floats, ints, longs, and strings. SQLite is lightweight database for which Android provides full support. It can be used for saving objects such as user, user-contacts, notifications etc. As default, it uses external storage which we have already discussed as not being secure enough. The best option to store our SQLite database is by using internal storage. According to Android documentation every application has its own internal storage area and only the owner of this area can use this storage; even owner of the phone cannot access it which means data can be saved securely. However, as we discussed before, there are already many existing violations of security system of Android. Therefore, trusting the system completely without taking any consideration is not feasible. However, keeping SQLite database in internal storage combined with some cryptographic functions would be the perfect solution to our problems. In user-profile, password is sensitive and it needs to be saved securely. Usually password is a long string that can contain different types of letters like integers, characters, etc. what makes it difficult for the user to remember it. The best approach to solve this problem is to use one more security layer which protects user password and is easy to remember: PIN. In general four digits long PIN is sufficient to achieve the required level of protection with some added limitations, like blocking of account after more than two times of entering the wrong PIN, to reduce the cases of offline/online attacks. Password will be used for registering the user with IM server and is needed during the login. It doesn't have to be user-dependent, since system requires it just one time and retrieves it from database every time when user wants to authorize himself by using PIN. Therefore, password can be generated randomly and it can be protected by using PIN provided by a user.

Verifying transactions and double-spending : Double-spending means successfully spending money more than once using the same means of transaction. While physical currencies don't have this problem, with digital currency there is a risk that the holder could make a copy of the digital money and send it to a merchant or another party while retaining the original. However, Bitcoin has a mechanism based on transaction logs which verify the authenticity of each transaction and prevent double-spending.

2.6 Push Notifications

Push notifications notify users about new messages or events even when users are not actively using the application. Notifications can be broadcast to only one, to several or only to registered users.

There are many comparisons between usages of Emails or SMS versus push notifications. Push notification approach is cheaper although it is more complicated. It is faster and spam rate is lower than Email and SMS systems. All in all, there are many benefits of using push notifications instead of Email or SMS.

With smart phones increasingly becoming ubiquitous, there is an increasing demand for feasible ways to notify users about new messages. Operating systems, such as IOS and Android, provide their own cloud-based push notification services. If we consider our selected platform, Android's push platform is still technically considered "beta" version, which means interfaces are evolving and they continue to change. This evolving nature makes direct integration of push notification platforms with user applications very complicated, since all changes have their own obstacles. There is no standardized usage. Furthermore, this dependence on other parties' services can make flexibility and control of the system harder.

Taking full advantage of supported and maintained platforms is a better approach. However, bugs and problems that come with every release of platforms might cause problems to maintainability across many systems. Current solution for that is building applications' own platform that provides notification services. However, building it with all specifications that Android Push platform provides might be extremely expensive.

2.7 PKI for Mobile Environments

In current mobile systems many applications provide end-to-end security by using public key techniques and an underlying public key infrastructure (PKI). This section examines and explains the basics of PKI and describes how they are used in current mobile systems.

All security mechanisms rely on either symmetric or public key cryptography. The combination of these two is being used in this project and provides both efficiency and security to the system. In order to understand the concept of PKI, we should first explain and compare these two cryptographic techniques.

2.7.1 Symmetric Key Cryptography

It is also known as Secret Key Cryptography. In this technique parties share a key that is used for encryption, decryption and also authentication. Security of this technique depends on keeping the symmetric key private. If an attacker steals shared symmetric key, he/she can read encrypted text and can have all rights of the real owner. Therefore, exchanging this shared key in secure way prior to the intended communication complicates the provision of security for transactions between entities that do not have a pre-established relationship.

Challenge-and-response method is one of the well-known methods that are used for authentication of parties. By using challenge-and-response one party presents a question ("challenge") and another party must provide a valid answer ("response") using the challenge and the secret key is the input for the algorithm. Challenger performs the same operation and compares the result with the received response. If the result and response are equivalent, only then the authentication succeeds.

2.7.2 Asymmetric Key Cryptography

It is also known as Public Key Cryptography and in contrast to Symmetric Key Cryptography, it uses different keys for encryption and decryption. In this technique each party has a pair of private and public keys and it is not possible to derive one from the other. As evident from their name, public key is publicly available while private key is private. There is no need to build a secure out-of-band key exchange since one of the keys is publicly available. However, distributing authentic public keys is an issue that needs an infrastructure. Digital Signature that is created by using private key and verified by public key is the way of authentication that proves possession of the private key. Unlike Symmetric Key Cryptography that encrypts, decrypts and authenticates with the same key, Asymmetric Key Cryptography uses different keys for encryption-decryption, signature-validation. Data is encrypted and validated with a public key and decrypted and signed with a private key.

While Asymmetric Key Cryptography has slow computation, Symmetric Key Cryptography has high speed computation for bulk data. Therefore it is impractical to use Asymmetric Key Cryptography to encrypt large amounts of data instead of Symmetric Key Cryptography. However, before the execution of symmetric algorithm, key exchange should be performed in a secure way. In practice they are often used together, so that Asymmetric Key Cryptography is used to encrypt a randomly generated encryption key and this random key is used to encrypt actual data using symmetric algorithm. This is sometimes called hybrid encryption.

Public Key Cryptography reduces the complexity of exchanging secret keys between parties. One party can create shared-key and encrypt it by using the other party's public key when (s)he wants to initiate a secure communication and sign it by using his/her own private key. Receiver then first obtains the secret key by validating and extracting the signature with sender's public key and then decrypts that information with his/her private key. However, it needs more sophisticated logic and additional organization for distribution of public keys when user-interactions increase like in case of handling certificates.

2.7.3 X.509 Certificates

Certificates are electronic documents that can be used as a proof of ownership of public keys. They include information about owner's identity, the key and digital signature of the entity known as Certification Authority and also includes information on who has verified the certificate's contents as correct. If the signature is valid, and the person examining the certificate trusts the signer, then that person knows (s)he can use that key to communicate with its owner. The certificate guarantees that the public key is bound to the entity whose Distinguished Name is included in the certificate. Certificate includes:

- a serial number that is unique number relative to the certificate issuer
- issuer name
- certificate owner's name
- public key of the owner
- algorithm used to calculate the signature

- expiration date that specifies the period for which the certificate is valid. Less validity signifies more security
- extensions which are optional and indicating how the certificate should be used.

2.7.4 PKI

Public key infrastructure is essential to manage certificates during their life-cycle and it has trusted third parties called Certificate Authority (CA) responsible for issuing and managing the status (expired, valid, invalid etc.) of certificates. According to article [1] there are six steps to manage the life cycle of the certificates:

1. Registration and key pair generation
2. Certificate generation and distribution
3. Certificate expiration
4. Certificate revocation
5. Certificate retrieval
6. Certificate validation

2.7.5 PKI in Current Mobile Systems

In general, security is achieved by using some service providers of second or third generation mobile networks, such as delivering smartcards with pre-installed symmetric keys that can be used to authenticate mobile devices. Trust relationship plays the main role when accessing network provider and the service provider for authentication, by taking a roaming agreement into the consideration. In case of providing confidentiality and integrity of data, symmetric session key is sent over the air. However, confidentiality and integrity across the entire path between two parties, i.e. end-to-end security, is not provided by these systems, and therefore has to be provided at application level.

There arises the need to have some assumptions for future usage of PKI in mobile systems, since currently there is no usage of PKI in mobile systems.

1. symmetric key provides more efficient computation; it consumes less energy and memory of mobiles;
2. non-repudiation is not a strict requirement for network access;
3. preshared secret key between the mobile node and the service provider can be installed in a relatively easy manner by implementing subscriber subscription procedure.

However, since there is still need for confidentiality and integrity of the symmetric key, applications should follow standard PKI mechanism explained above.

3 Security Architecture for Bitcoin Transactions

3.1 System Components

In this section we examine all system components shown in Figure 2 with their relations and access points.

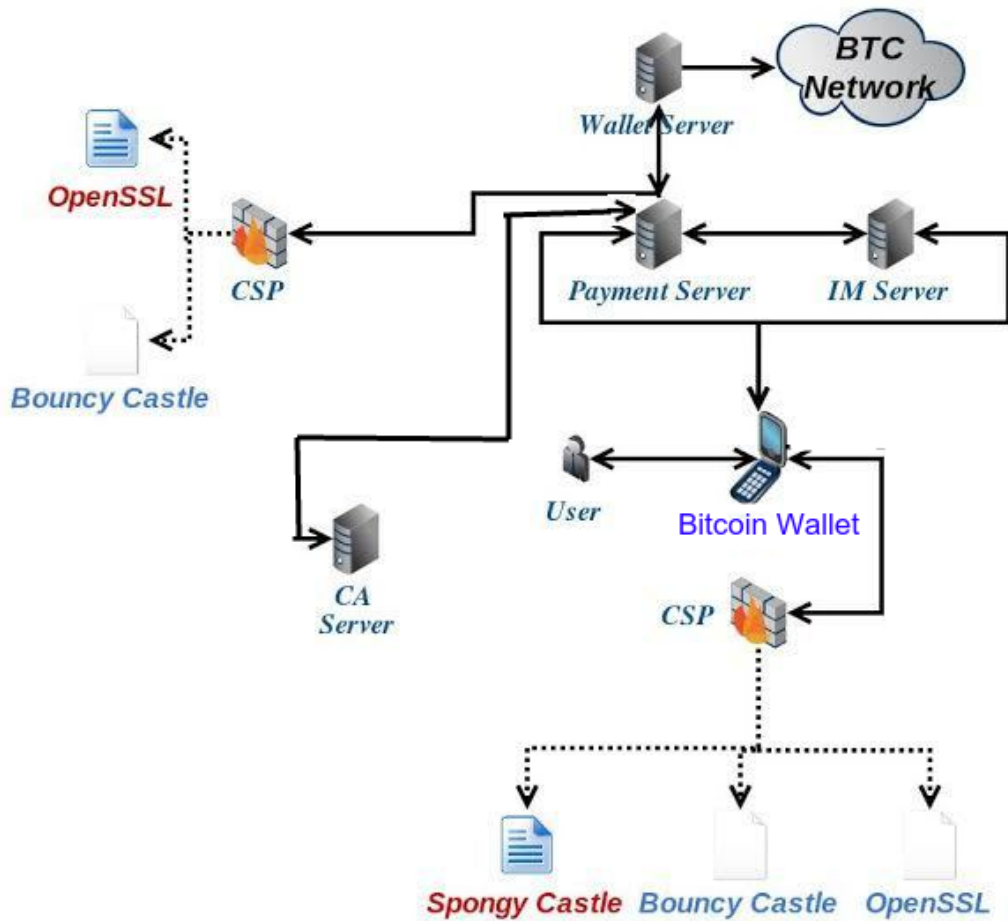


Figure 2: System Components

3.1.1 Crypto Service Provider (CSP) [18, 20, 21]

Crypto Service Provider is a unified software library which has full crypto functionalities, supports all standard crypto algorithms, provides crypto packaging formats and uses alternative crypto engines. CSP represents the lowest level of our system. CSP is available for multiple platforms, such as PC/Windows environments and for mobile platforms. Whichever alternative software library it uses, all its functions have the same function signature which makes it standardized. Changing upper layer platform (switching from PC to mobile platform) will not require modification of the application code.

In this project CSP uses Spongy Castle API for Android client applications and OpenSSL for Windows based Payment Servers as alternative crypto libraries.

OpenSSL : OpenSSL is an open-source implementation of the SSL and TLS protocols. It is written in C programming language and implements basic cryptographic functions and provides various utility functions. It supports most Unix-like platforms including Solaris, Linux, Mac OS X, Microsoft Windows and Android.

Bouncy Castle : Bouncy Castle is a collection of APIs used in cryptography. It contains almost all well-known APIs including OpenSSL API. It is written in Java and also in C#. Its architecture comprises two main components that support basic cryptographic capabilities: light-weight API and JCE (Java Cryptography Extension) provider. Its flexibility and powerful architecture used by many different APIs were the main reason to use it in our Android System. Compatibility issue was also considered for this choice.

Spongy Castle : Spongy Castle API is the repacked version of Bouncy Castle API for Android applications. Because of its size, Android platform includes Bouncy Castle in a crippled state. Only part of its initial functionality is included because of size constraints. It also makes installing an updated version of the libraries difficult due to classloader conflicts. Therefore, Spongy Castle is the collection of Bouncy Castle libraries with a couple of small changes to make it work on Android without any conflicts with the embedded Bouncy Castle API in Android.

3.1.2 Client Application

Application represents the second layer of the system. It can communicate interchangeably with User, CSP, Payment Server, Instant Messaging Server and CA Server. It is an Android based client application written in native language of Android: Java.

3.1.3 User

User is a person who wants to use Bitcoin Wallet to receive and to make payments. For that, they need to use the Client Application to create their accounts and addresses associated with those accounts.

3.1.4 Instant Messaging (IM) Server

IM Server is one of the main components of the Secure Bitcoin Wallet system installed on either Microsoft Windows or Linux based computers. It uses Openfire (Section 2.3)

server as an underlying system. It provides instant messaging service to the user and also it is used as a notification server in our system for informing payee and payers about incoming and outgoing transactions.

3.1.5 Payment Server

Payment Server is one of the main components of the Secure Bitcoin Wallet system installed on Microsoft Windows computers. It is responsible for all Bitcoin transactions, such as "Request Payment", "Send Payment", "List Balance and Transactions", "Registration", which will be explained later. It can communicate interchangeably with Client Application, IM Server, and Wallet Server. It uses batch files to execute pre-defined Bitcoin transactions and sends these batch files to Wallet Server to complete transactions. It also informs parties (payer and payee) accordingly about the reply from the Wallet Server.

3.1.6 Wallet Server

Wallet Server is a daemon that executes all transactions. This software is open-source and can be run on PC. It takes batch files created by Payment Server, processes the request and execute the command related to the transaction. It maintains of all user's accounts and contents of user wallet. Basically, wallet is just a digital file ledger that contains user account info, such as Bitcoin addresses, balance transactions etc. for each user. Whenever user wants to perform payment, the Wallet checks balance of the sender and if balance is greater than the transaction, it completes transaction and sends digitally signed messages to the network BTC Network by using broadcasting. Messages are broadcast on a best effort basis. Transactions are recorded in a distributed public database known as blockchain, with consensus achieved by a proof-of-work system called "mining". Blockchain is distributed internationally using distributed broadcast protocol.

3.1.7 Bitcoin Network

Please check Section 2.4.

4 Sytem Protocols and Operations

4.1 Local and Remote Registration Protocol

One of the most important aspects of "Registration" operation is to solve the security vulnerability of "User Service" plugin explained in Section 2.3. One solution to that problem could be modifying plugin and enabling https authentication. However, since it is open source and there is opportunity to modify it, isolating either secret key or administrator's password and name the outside access is deemed to be the best and fastest solution. Therefore, IM Server and Payment Server run on the same machine and Payment Server is responsible for registration. Connection security between all components relies on the security of SSL at this point.

Figure 3 explains steps and flow of messages in system during registration.

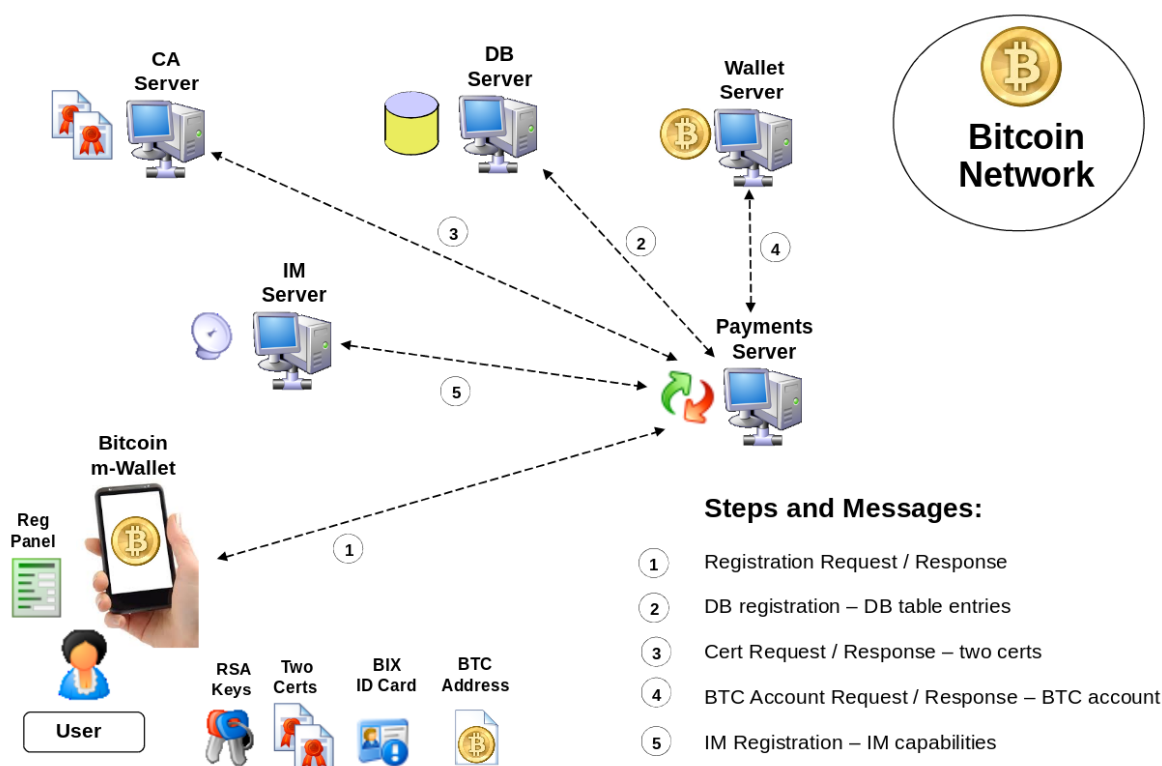


Figure 3: Components and Protocol for User Registration

Protocol follows the following steps. All parties perform their tasks successfully or produce an error message. Hence, the protocol always completes.

1. User installs the application into her/his Android mobile phone.
2. During the first activation, he/she is prompted to enter her/his mobile phone number, email, password, photo, country code, mobile number and PIN. User only has to do this process once. Each subsequent time when the user uses the application, he/she will only need to enter her/his PIN.

3. Client application will create user's name automatically by using user email address which is unique and which will provide unique name for the user. System then creates two RSA keys by using CSP module and creates certification request by using its public keys. At the end, user will be registered to Payment Server by using encryption of user details such as user name, password and user email address, county code, mobile number and certification requests. Connection between Payment Server and Client Application is based on SSL security.
4. Payment Server first decrypts the user data by using Crypto Service provider and adds the user to the database server which actually run on the same machine as the Payment Server. Then Payment Server sends user's certificate requests to the CA server and receives two certificates as a response if authenticated.
5. Payment Server then connects to Wallet Server which run also on the same computer and makes BTC Account Request and gets BTC Account as a response.
6. For the last step Payment Server connects to IM server through SSL connection and registers user with the IM Server by using user name, password and email address.
7. If everything goes well and the Payment Server completes its tasks described above, it responds to the user about registration with two certificates.
8. After user registers successfully into the system, for local authentication hash of user's PIN will be encrypted by using user password. User password will be encrypted by using the PIN provided by user. BIX ID card consisting of user name, email, encrypted password, photo, first and last name and encryption of PIN's hash will be created and stored into the SQLite database on internal storage of user mobile phone.

4.2 Local and Remote Authentication Protocol

After user completes registration user is granted permission to access the system by using his PIN defined in the registration step. Login credentials are transmitted by using SSL connection between Payment Server and Client.

Figure 4 explains flow of message in the system during the registration process.

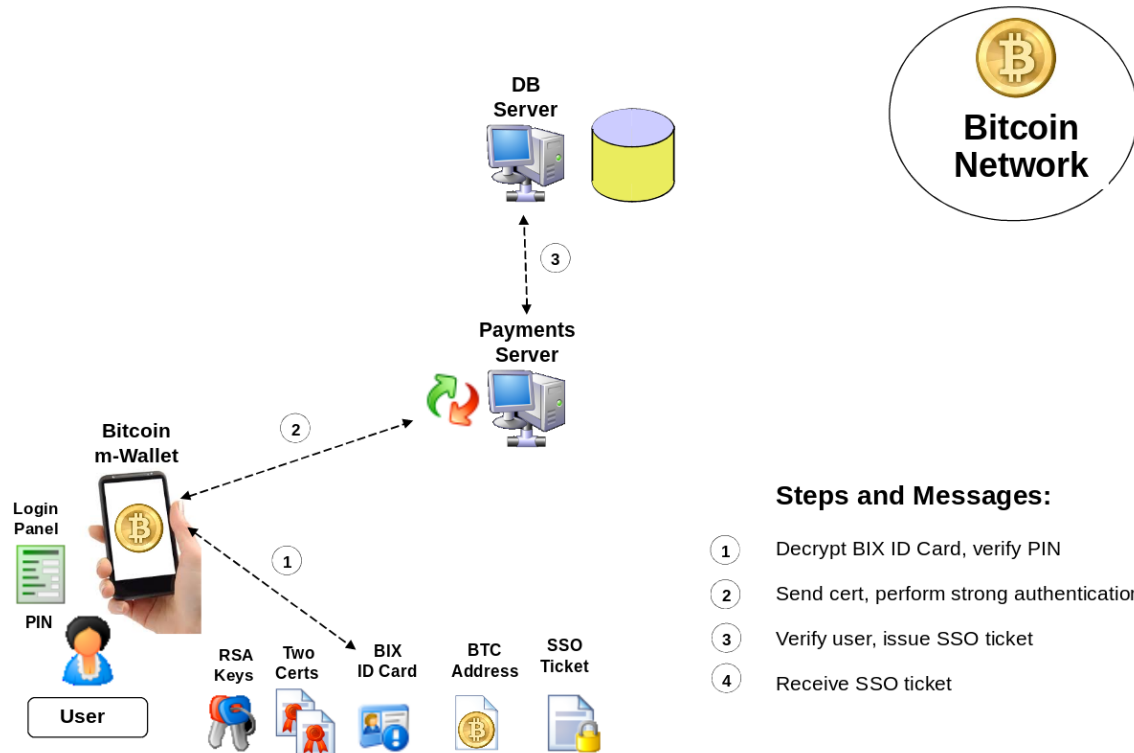


Figure 4: Components and Protocol for Local and Remote Authentication

The protocol follows the following steps. All components perform their tasks successfully or produce an error message. Hence, the protocol always completes.

1. Client application gets user's PIN and accesses database stored on internal storage to decrypt the encrypted password. System first decrypts user password by using given PIN and uses decrypted password to decrypt hash of user's PIN. Then it hashes the given PIN and compares it with the decrypted hash stored on the internal storage of user mobile phone. If both are validated to be the same, user will have access to the consecutive processes.
2. Application accesses Payment Server by using user's name, decrypted password and its certificate through SSL connection.
3. Payment Server will verify the user by using received certificate and also password of the user. If PIN is correct, decryption of encrypted password will give the actual password which means that login attempt is successful. Otherwise, attempt of user to login will fail.
4. If the user is the same person that he is claiming to be, Payment Server will send user Single Sign on ticket.

4.3 The Certificates Protocol [22]

After securely generating public/private key pair, the client creates certificate request and connect with Payment Server. Payment Server makes X.509 Certificate request to the CA. The request contains unique identity which is transferred in the form of certificate for user. Explanation about how to make a request for X.509 certificate from the CA can be found in RFC 4211 in detail. Typically the request includes public key and entity's distinguished name and other required information. RFC 4211 recommends some steps to generate certification request:

1. A CertRequest object should include the public key, subject name and other requested certificate fields related to the registration process depending on the chosen CA's Certificate Request Policy (CRP).
2. If required, private key's (corresponding to the public key for which a certificate is being requested) proof-of-possession value is to be calculated and it should be attached in advance with certificate request message subject.
3. Entity should sent certificate request message securely to CA. Base64 encoded format should be used as the message format, so that CA can easily understand it.

POP means that the CA is adequately convinced that the private key that entity keeps is associated with the respective public key. POP aims to prevent common attacks such as non-repudiation of transactions and to allow CA to properly check the validity of the binding between the subject and the key pair. CA's Certificate Request Policy is defined in POP according to usage of the certificate i.e. signature, encryption or key agreement key pairs.

CA responds to the Certificate request by digitally signing the public key certificate with its private key and then CA publishes and sends the generated certificate which is signed to the requested entity.

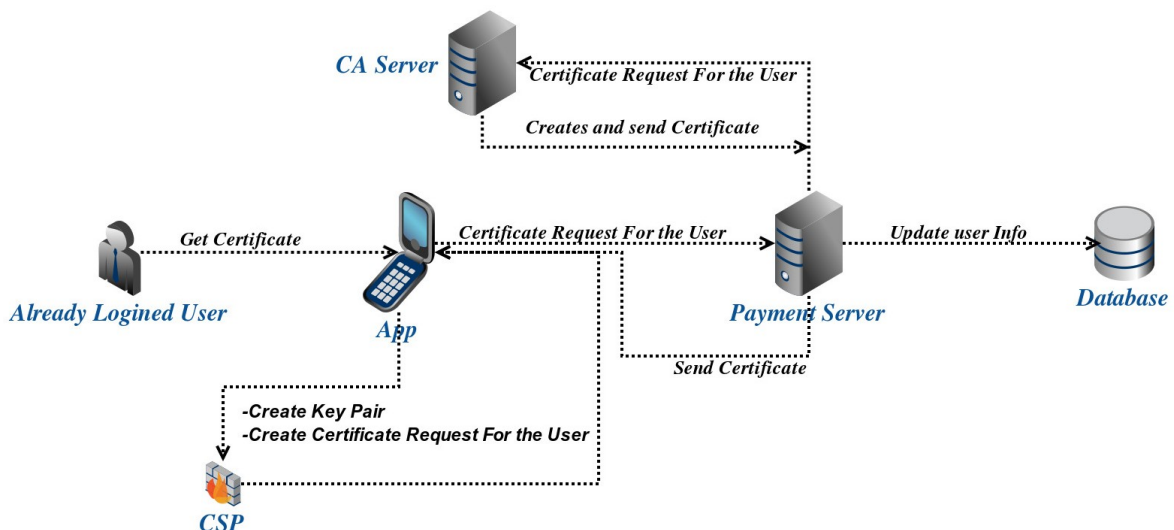


Figure 5: Protocol to Request/Receive Certificate

4.4 Notification Messages Protocol

This functionality is not available to user if he/she does not have a certificate. Furthermore, in the login step user is getting login to the system through the Payment Server and he/she is offline at the IM Server until he/she enables notifications. Others can send messages to him/her, but he/she will not be able to view these messages until the user enables notifications. Figure 6 explains flow of messages in system during the process of enabling notifications.

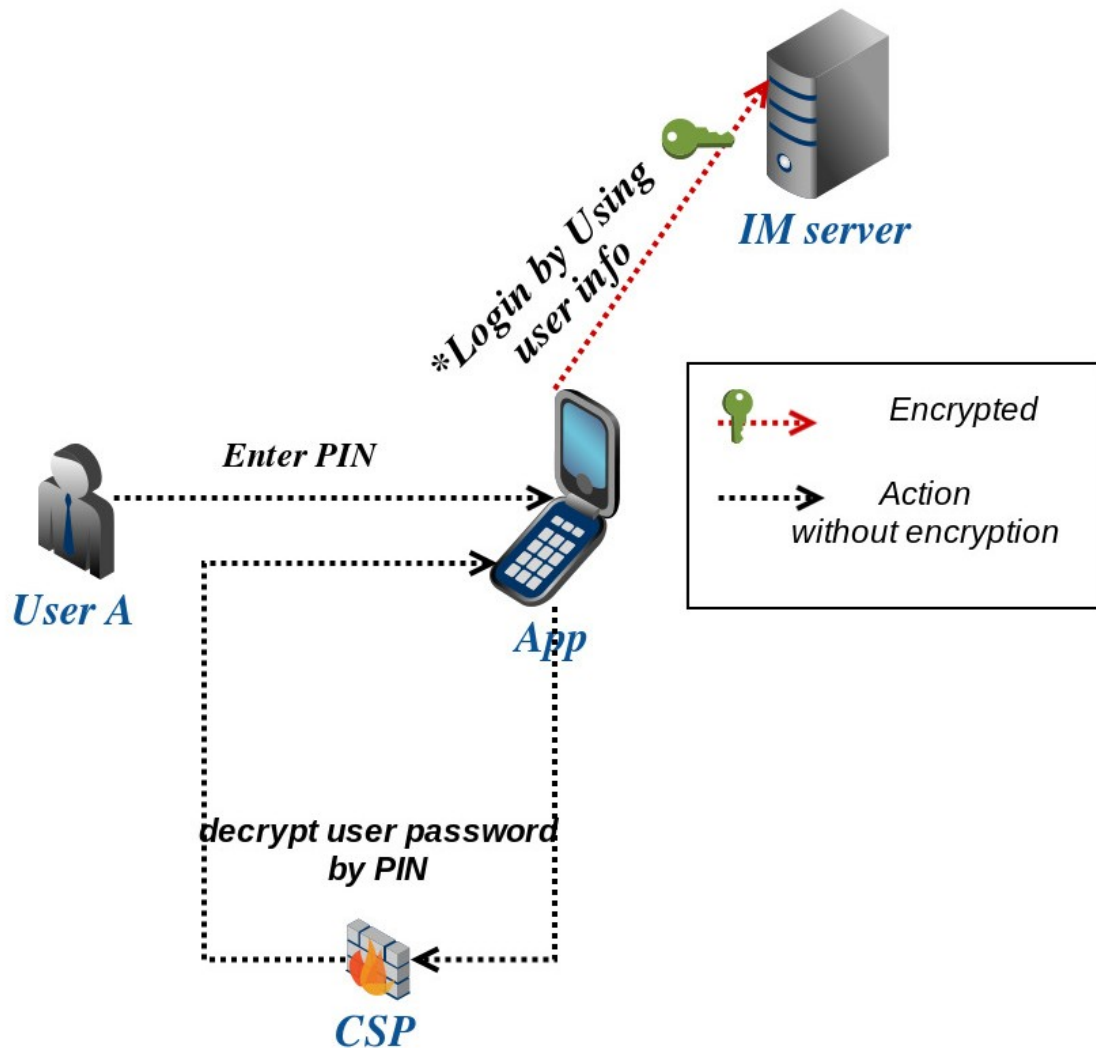


Figure 6: Notification Messages Protocol

In order not to bother users with notifications, system's notification feature is disabled as default. Therefore, whenever user wants to enable the notifications he/she should follow these steps:

1. User should click "Enable Notification" button and client application asks PIN of the user and decrypts user password by using CSP.
2. System accesses IM server by using user's name and decrypted password through SSL connection based on XMPP.

3. If PIN is correct, decryption of encrypted password will give the actual password which means that login progress is successful. Otherwise, attempt of user to login fails.
4. Lastly, user will be connected to the IM server until he kills the application.

4.5 Instant Messaging Protocol

This functionality is not available to the user if he/she haven't had a certificate and if he/she hasn't enabled notifications. All communications going through IM Server and security of the communication relies on security of CSP and SSL is provided by IM Server. Figure 7 explains flows of messages in the system while sending and receiving instant messaging services. Protocol follows the following steps. All components perform their tasks successfully or give some error. Hence, the protocol always completes.

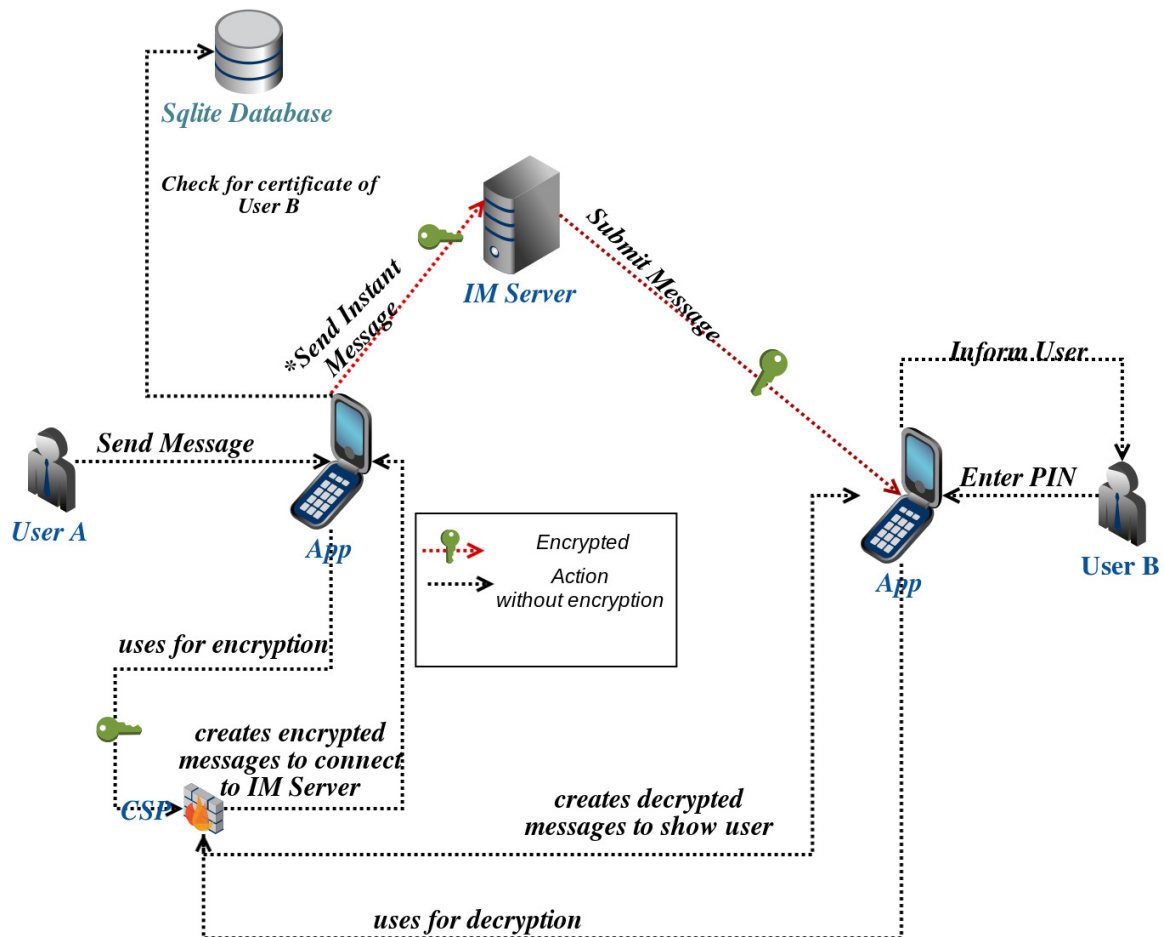


Figure 7: Instant Messaging Protocol

1. The client application enables user to send and receive instant messages in a secure way.

2. User should click "Secure IM" and choose "Send Message" option.
3. User should enter receiver's email address and message text.
4. System checks if the receiver's email has correct email format or not and then checks whether it has a receiver certificate in the local database.
5. If the system cannot find certificate for the intended receiver in the database, it creates one message and sign it with the private key and attach its certificate to the end of the message.
6. If it has receiver's certificate, it uses that certificate to encrypt and sign the message.
7. In both above cases, system sends message to the IM Server and IM server transfers that message directly to the receiver.
8. Whenever a sender gets message, the system check for the message tag. If it starts with <certificate> and ends with </certificate>: The certificate of the sender is given and the message is validated. If the message is authenticated to be send by the sender system, sender's certificate is added to the database and receiver's certificate is sent to sender. When the sender receives the certificate of the receiver, they can start to use secure instant messaging service. If it doesn't start with indicated certificate, system pulls out sender details from database to validate and decrypt this message.
9. If users don't have certificates of each other, they cannot communicate and they cannot also do any payment transactions. This approach solves non-repudiation and security problems.

4.6 Listing Balance and Transactions Protocol

This functionality can not be used by user if he/she does not have a certificate. Security of this step relies on SSL connection. Figure 8 explains flow of messages in the system during the requesting and receiving BTC balance and transactions.

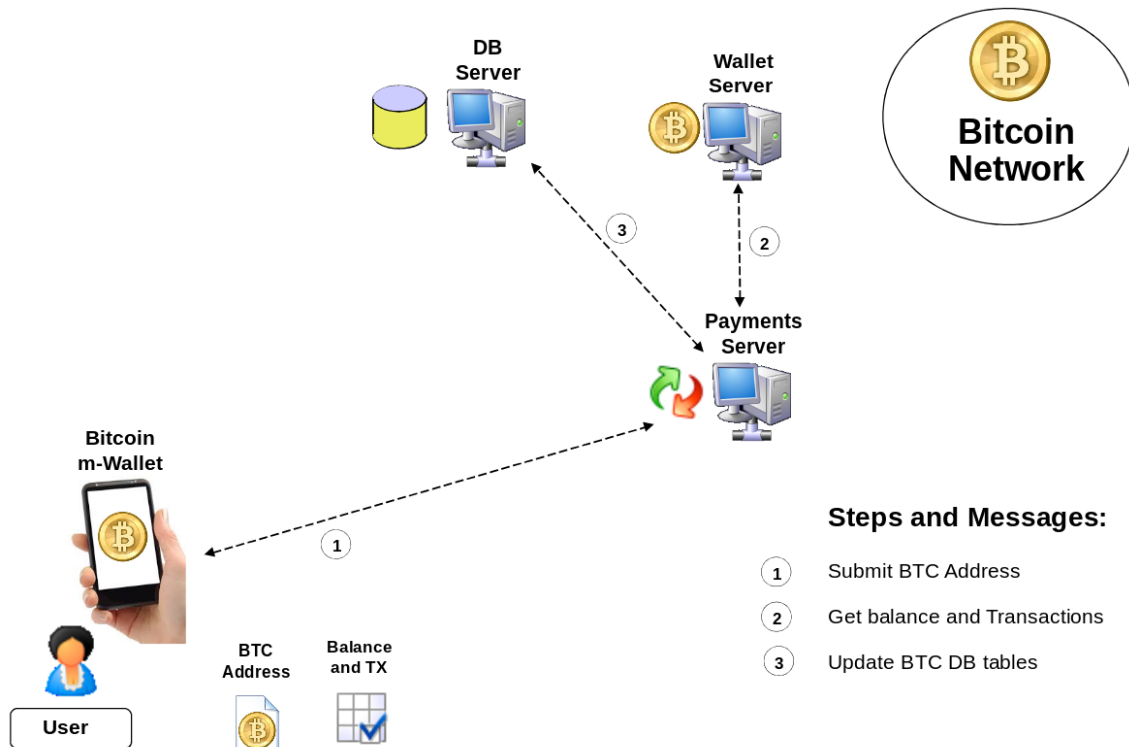


Figure 8: Components and Protocol to List Balance and Transactions

Whenever user wants to list balance and transactions he/she should follow these steps:

1. The client application includes functionality to check users' balance and previous transactions.
2. The application connects to the Payment Server to retrieve balance and previous transactions through SSL connection.
3. Payment Server sends the required information data back to the client.
4. The application shows balance and previous transactions.

4.7 Requesting Payments Protocol

This functionality can not be used by user if he/she doesn't have a certificate. This step is not much different from instant messaging. All steps are the same. All communications go through Payment Server and security of the communication relies on security of CSP and also SSL is provided by IM Server.

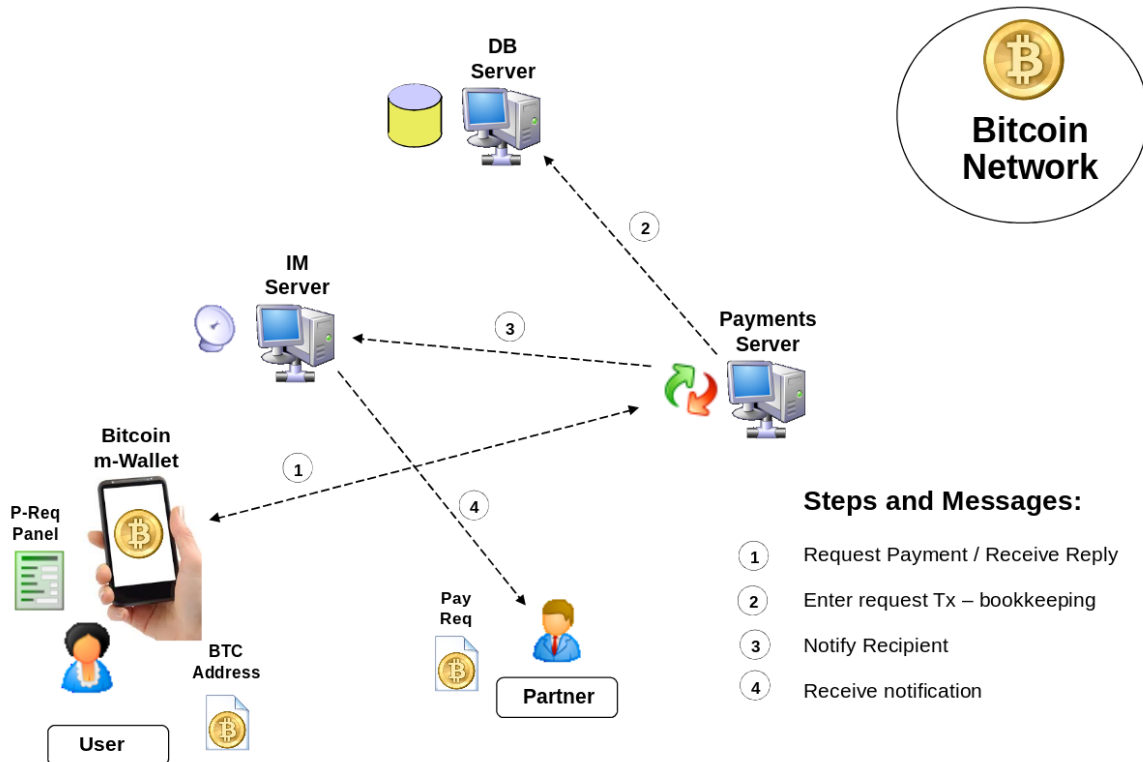


Figure 9: Component and Protocol for Requesting Payments

Figure 9 shows flow of messages in the system during requesting payments. The protocol follows the following steps:

1. User can request payment by using QR code or send the request directly to anyone by using receiver's email
2. User enters email address of the merchant manually or use QR code to take embedded email address of the merchant.
3. System checks if the receiver email is in the appropriate format and further checks to make sure the receiver certificate is present in the local database.
4. If the system cannot find the certificate for the indicated receiver in the database, it creates one message and sign it with its private key and attaches its certificate to the end of the message.
5. System sends that message to the IM Server and IM Server transfers that message directly to the receiver.

6. When the payer receives a message, it checks message tag. If it starts with <certificate> and ends with </certificate>: it gets certificate of the payee and validates the message. If the message is sent by the payee, the system adds payee's certificate to the database and sends receiver's certificate to the payee. Whenever payee gets certificate of the receiver, payee can do Payment Request.
7. Client encrypts user's request by using receiver's public key embedded in receiver's certificate and signs it by using his private key.
8. The application sends securely packed requires of the user to the IM server.
9. IM server sends this request to the payer.
10. When the payer gets this request, his/her system validates and decrypts the request by using CSP.

4.8 Making Payments with Requests Protocol

This functionality can not be used by the user if he/she doesn't have a certificate. It is complementary to the request payment.

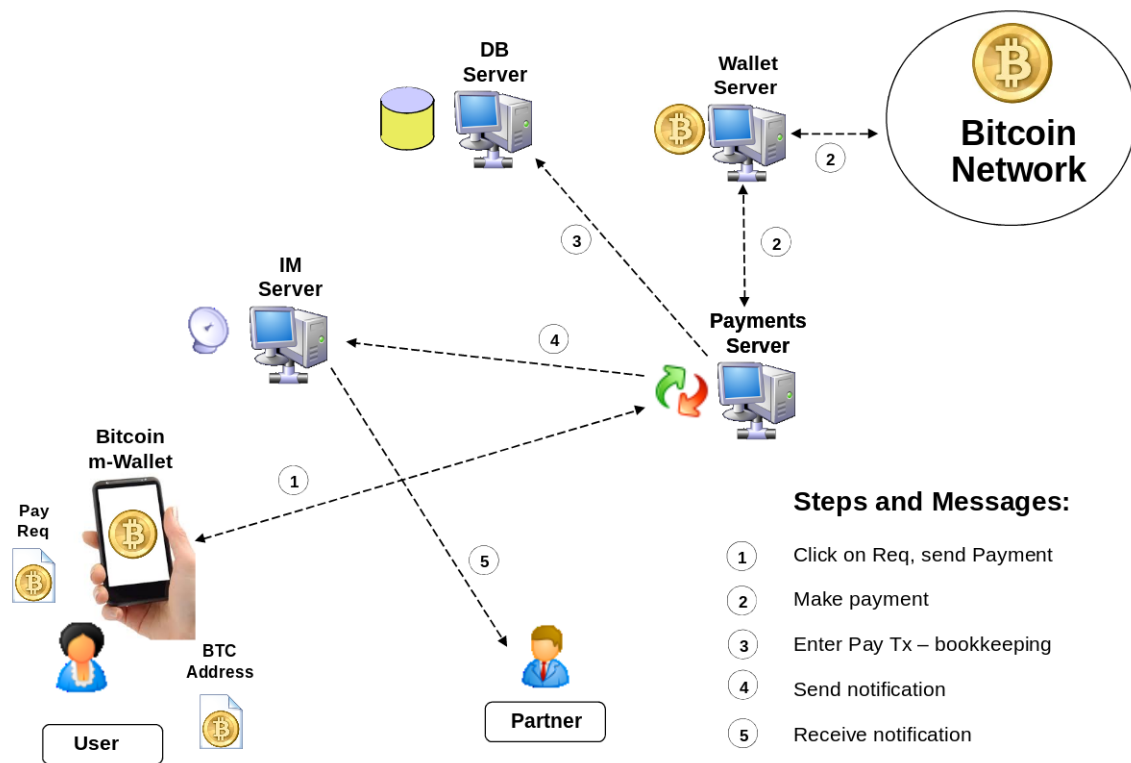


Figure 10: Componential Protocol for Making Payments(with Request)

Figure 10 shows flow of messages in the system for making payments with request and protocol follows the following steps:

1. The application notifies the user whenever he/she gets new payment request. User also can retrieve all requests later by using "Listing Balance and Transactions" functionality.
2. User clicks the notification or required item in transactions section.
3. System shows the details about request including payee identity and amount.
4. User confirms the payment and clicks "Send".
5. The application gets entries of user and sends it to the Payment Server which takes instructions from the client application and executes the batch files of Bitcoin CCL.
6. Payment Server creates instructions and changes it to the executable batch file. Batch file has userID of the receiver and sender with payment amount.
7. Payment Server sends this batch file to the Wallet Server.
8. Wallet Server processes this batch file and retrieves receiver's and sender's userID.

9. It retrieves Bitcoin addresses associated with those userIDs from its database.
10. Wallet server checks the balance of sender's account. If sender has enough Bitcoins, it completes the payment and updates sender's and receiver's accounts.
11. When the Wallet Server completes the required action, it sends a feedback to the Payment Server to inform the BTC networks to prevent double-spending.
12. Payment Server gets feedback from Wallet Server and connects to IM server to inform both parties about the outcome of the transaction.
13. IM server sends confirmation messages to both parties.

4.9 Making Payments (without Request) Protocol

This functionality can not be used by user if he/she doesn't have a certificate.

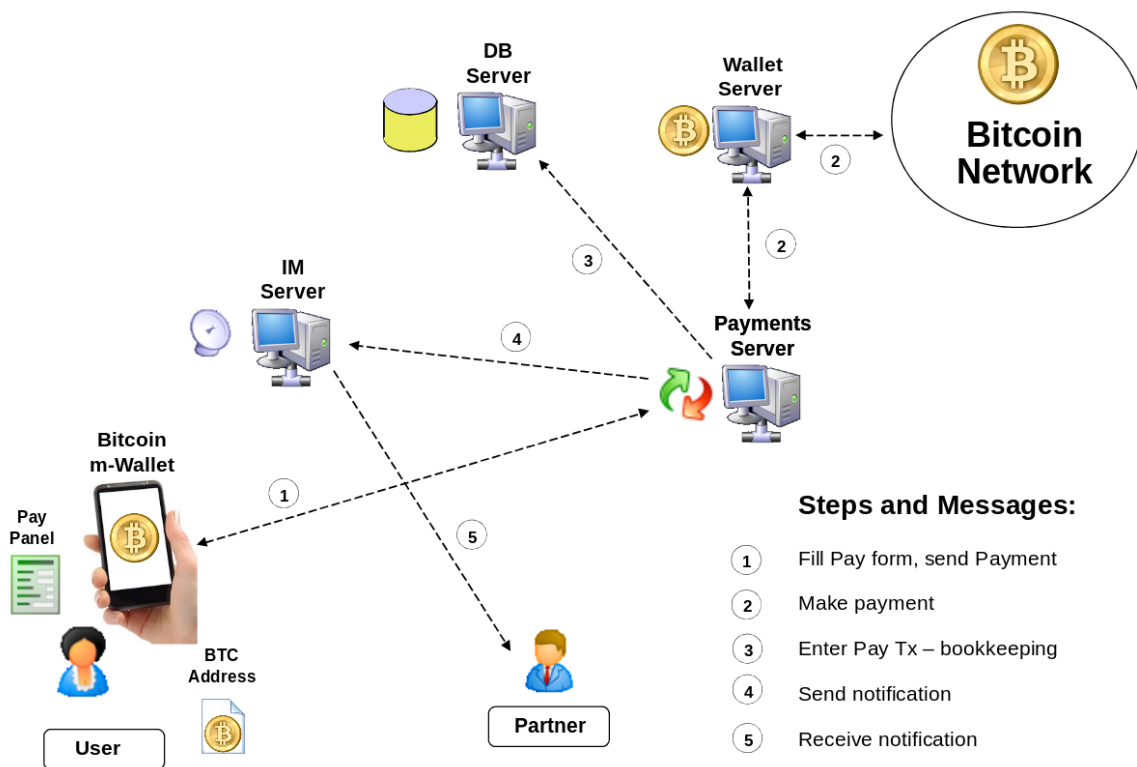


Figure 11: Components and Protocol for Making Payments(without Request)

Figure 11 shows flow of messages in the system during making payments without requests. The system follows the following steps:

1. The application also offers service to the user to send payments by using receiver's email with indicated Bitcoin amount
2. User enters email address of the receiver and the amount of BTC that he/she wants to send.

3. The application gets entries of user and sends it to the Payment Server which takes instructions from the client application and executes batch file of Bitcoin CCL.
4. Payment Server creates instructions and changes it to the executable batch file. Batch file has userID of the receiver and sender with also the payment amount.
5. Payment Server sends this batch file to the Wallet Server.
6. Wallet Server processes this batch file and retrieves receiver's and sender's userID.
7. It retrieves Bitcoin addresses associated with those userIDs from its database.
8. Wallet Server checks the balance of sender's account. If sender has enough Bitcoins, it completes the payment and updates sender's and receiver's accounts.
9. When the Wallet Server completes the required action, it sends a feedback to the Payment Server to inform the BTC networks to prevent double-spending.
10. Payment Server gets feedback from the Wallet Server and connects to the IM server to inform either both the parties or just the sender on the outcome of the transaction. If transaction is successful Payment Server inform both parties, otherwise it informs just sender with error.
11. IM Server sends message either to both parties or just to sender depending on the message that it received from the Payment Server.

5 Design of the Demo

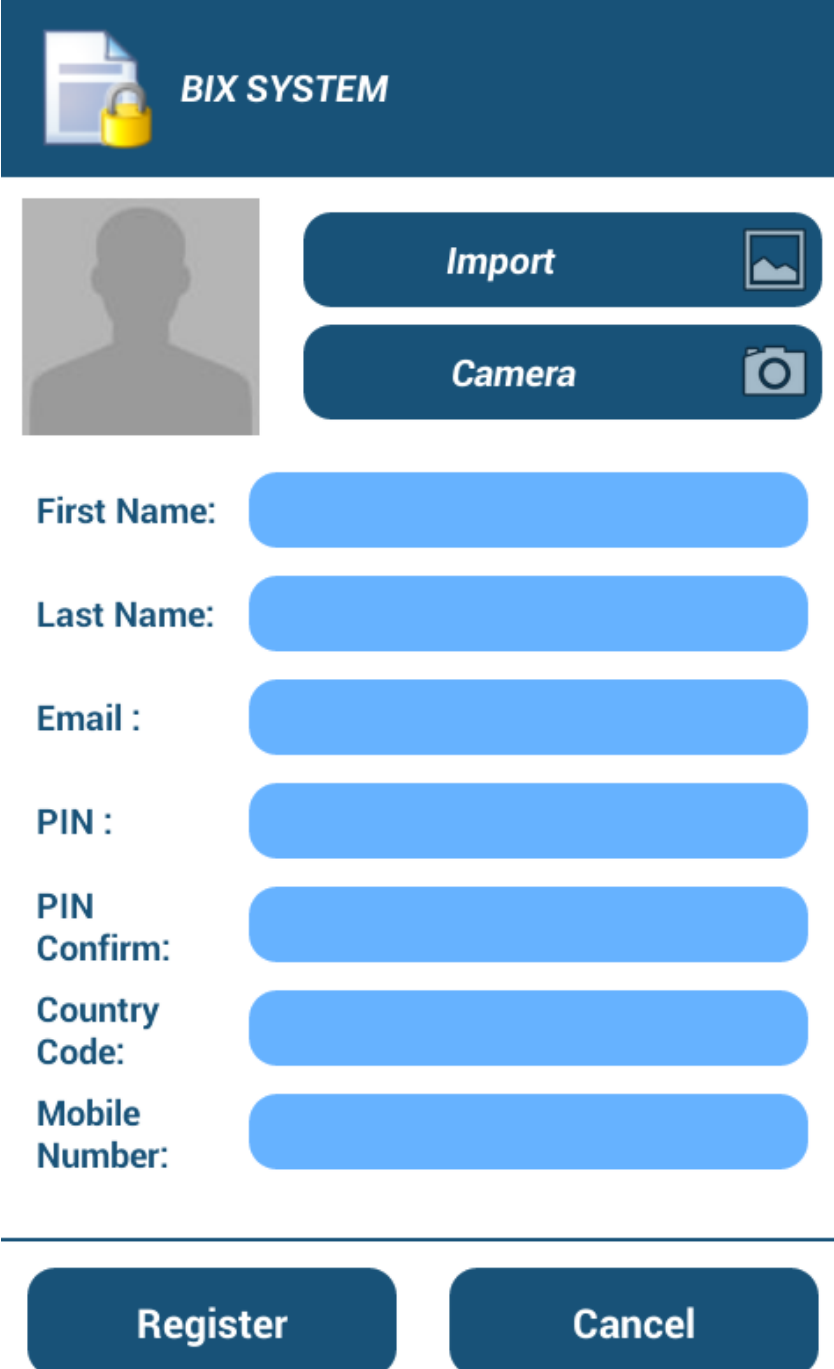
Only Symmetric Key Cryptography was used for this demo.

5.1 Launching

The client application is unique for each user. In one mobile phone, only one user profile can be stored. Therefore, whenever user launches the client application, the system checks if user's profile exists or not in local storage of user's phone. In technical perspective, system checks whether a SQLite database exists on local storage or not. If the user still hasn't signed up, the system will not be able to find database and user will start by registration; otherwise he/she will continue with the login step.

5.1.1 Registration

With the first activation user should register himself/herself to the system. If user doesn't have account defined in its current phone system, Figure 12 is shown :



The registration panel features a dark blue header with a document icon and a yellow padlock, labeled "BIX SYSTEM". Below the header is a grey silhouette of a person. To the right of the silhouette are two buttons: "Import" with a picture icon and "Camera" with a camera icon. Below these are seven input fields: "First Name:", "Last Name:", "Email :", "PIN :", "PIN Confirm:", "Country Code:", and "Mobile Number:". At the bottom, there are two buttons: "Register" and "Cancel".

Figure 12: Registration Panel

5.1.2 Registration Errors

In order to complete registration user should fill out all fields that application requires. Otherwise, the system produces an error message, shown in Figures 13, 14 and user can not continue:

The image displays two side-by-side screenshots of a registration form titled "BIX SYSTEM". Each form has a header with a logo and the text "BIX SYSTEM". Below the header, there is a profile picture placeholder, an "Import" button with a camera icon, and a "Camera" button with a camera icon. The form contains several input fields: First Name, Last Name, Email, PIN, PIN Confirm, Country Code, and Mobile Number. Each field has a red exclamation mark icon indicating an error. In Figure 13, the First Name field is empty, and a red-bordered error message box points to it with the text "First Name field can not be empty!". In Figure 14, the Last Name field is empty, and a red-bordered error message box points to it with the text "Last Name field can not be empty!". At the bottom of each form, there are two buttons: "Register" and "Cancel".

Figure 13: Registration Error 1

Figure 14: Registration Error 2

5.1.3 Login

Figure 15 shows "Login" page of the system. After user registers himself/herself once on his/her current phone he needs to login to the system whenever he/she wants to access the system.

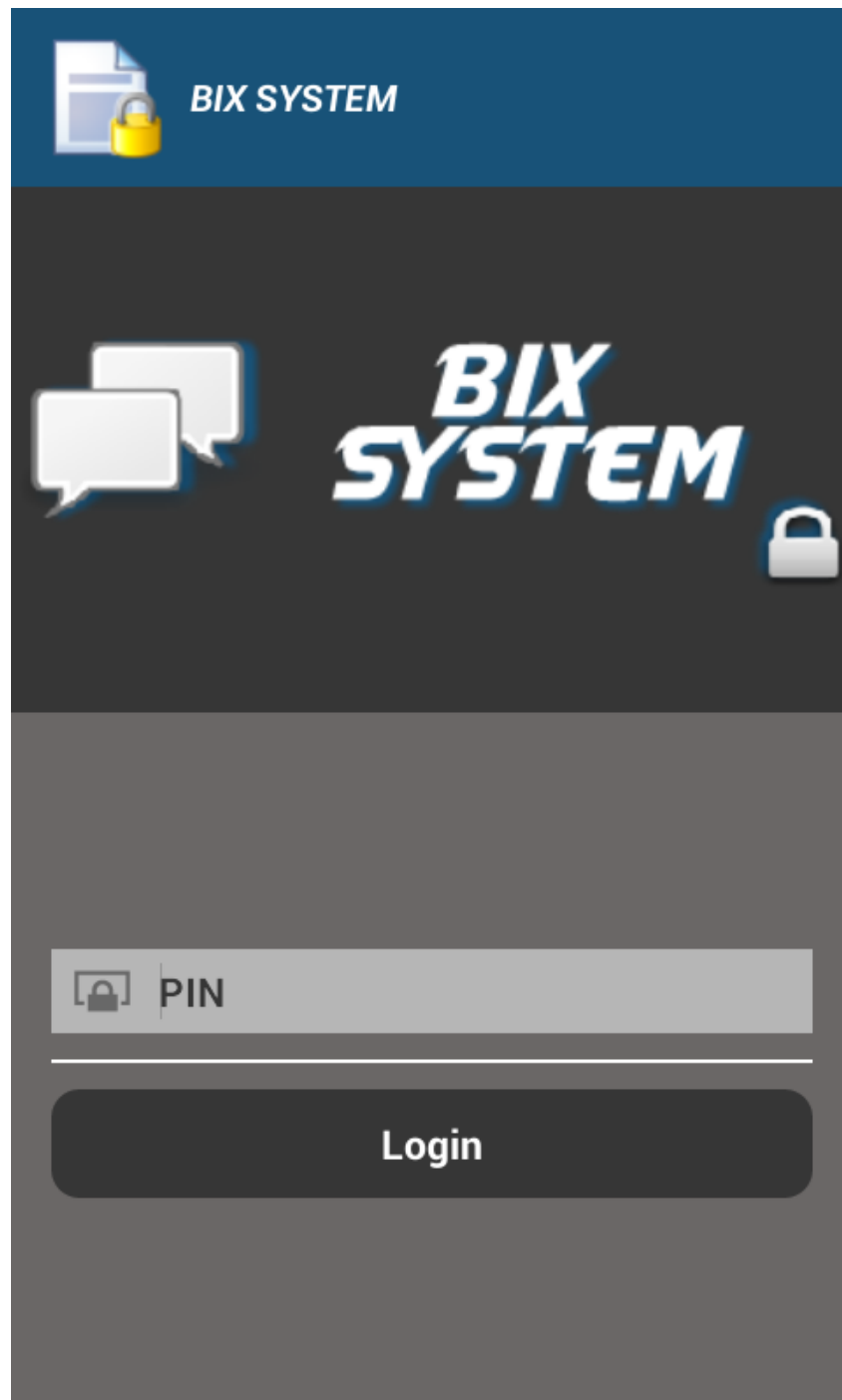


Figure 15: Login Panel

5.2 Main Page/Main Menu

After user attempts to login just by using PIN that he/she had defined during the Signup step, system decrypts password of the user by using that PIN and connects to the IM server to try login. If PIN is correct, remote authentication will be successful and also local authentication. If authentication is successful, system will show main page with ID card as default (Figure 16) option with navigation drawer layout containing left side menu (Figure 17).

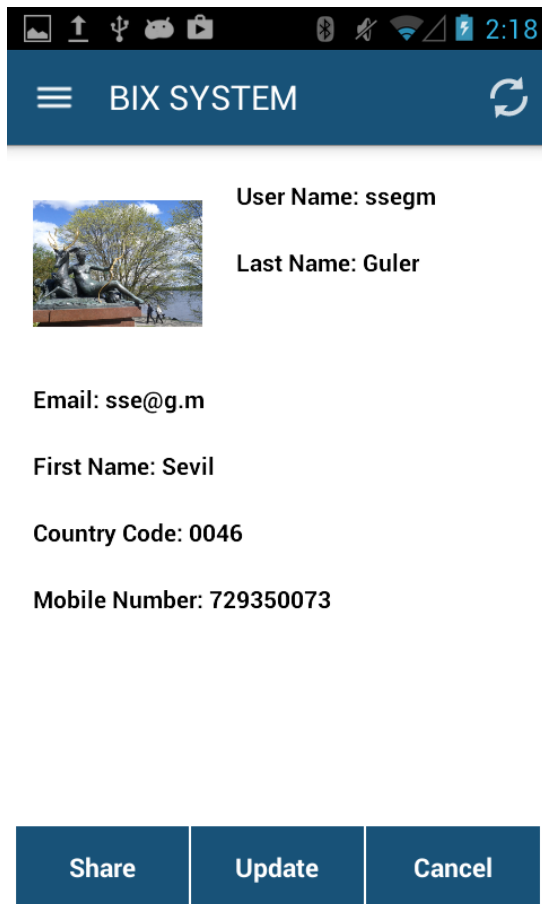


Figure 16: ID Card

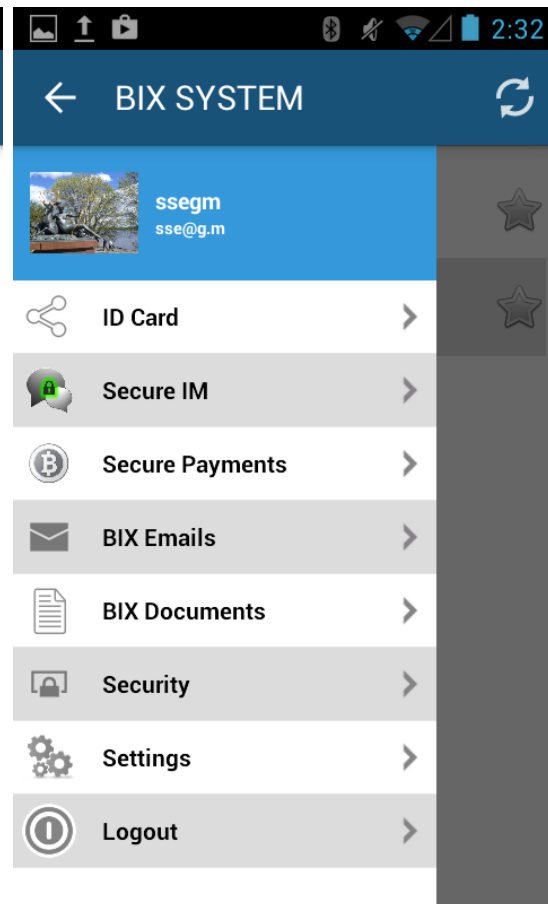


Figure 17: Main Menu

5.3 Secure IM System

The client application offers also secure instant messaging service. User can fetch his/her previous or current messages (Figure 18) or he/she can send and receive messages by entering receiver's E-mail address (Figure 19). System will trigger the receiver for either offline or online messages whenever he/she is online.

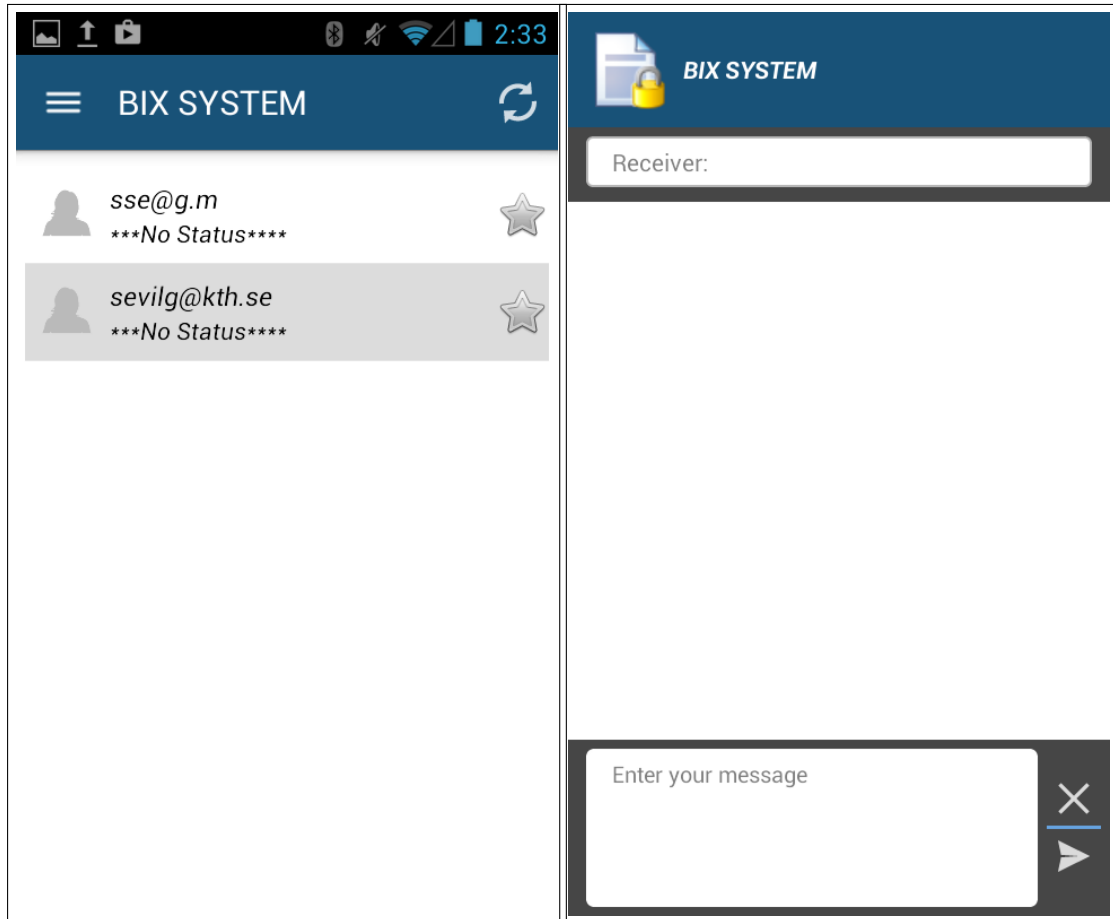


Figure 18: Fetch Message Panel

Figure 19: Send Message Panel

5.4 Secure Payments

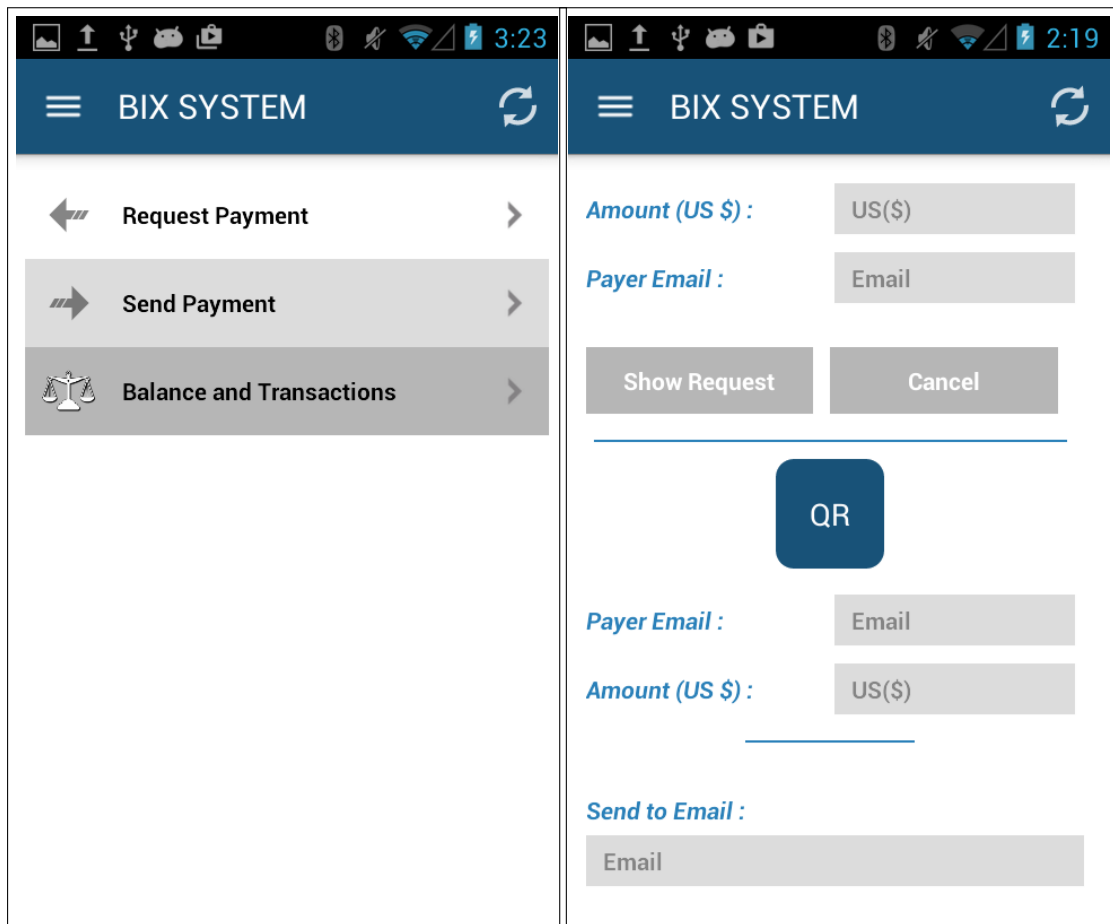


Figure 20: Secure Payments Panel

Figure 21: Request Payments Panel

This is the main aspect of the project: Secure Payments. You can find workflow in Chapter 3. By default, user will see menu (Figure 20) and click on "Request Payment" (Figure 21), on "Send Payment" (Figure 22) and on "Balance and Transactions" (Figure 23) will be shown.

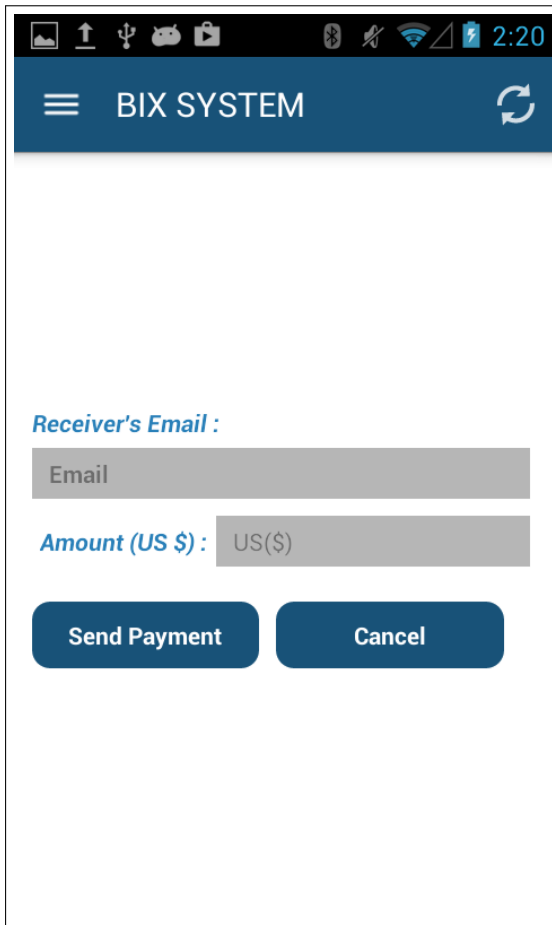


Figure 22: Send Payments Panel

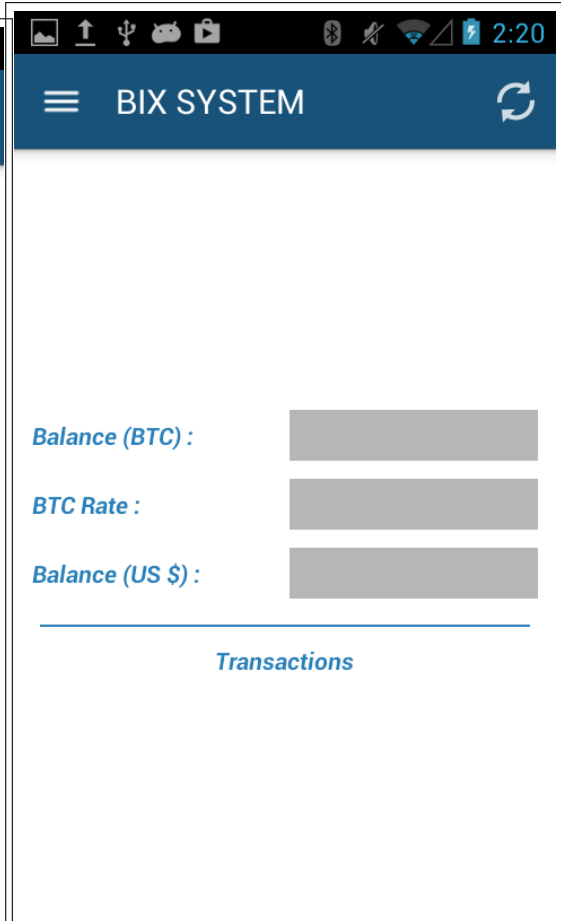


Figure 23: Balance and Transactions Panel

6 Evaluation of the System using Engineering Standards

In this section "Secure Bitcoin Wallet" is evaluated by using some engineering standards. Risk and cost management and analyzing the quality of the Wallet were one of our main aims of the project.

6.1 Scope

Scope of the "Secure Bitcoin Wallet" is for all users of Android mobile phones who want to use Bitcoins as a mobile payment in a secure way with high reliability and usability.

6.2 Economy

This project is written by using Java and with an open source approach. It is free and it can be used and modified by another Android developers. From the user perspective, users are connected to the system through the Internet which is quite cheap compared to other means of communication.

6.3 Reliability

Application guarantees security and privacy of users. The system was thoroughly tested and found reliable.

6.4 Usability

Application is easy to use. It targets not only people with a computer science background but also general users.

6.5 Performance

Performance of the application strictly depends on speed of the internet connection. Application is responsive and fast. All of the time consuming operations are executed via threads instead of UI thread.

6.6 Objectives and Success Criteria of the Project

Application achieves encrypting/decrypting/signing all sensitive data and guarantees secrecy and security of the user.

6.7 Ethical

As a business end user, it is our responsibility to promote ethical uses of information technology in the workplace. Therefore, any information provided by user is not being used for any purposes. Privacy is a right of everybody. Project gives a great chance to all users to protect privacy of their transactions.

6.8 Security

One of the main goals of this project was providing security to all users. User authentication should not fail and digital currency should not be forged, or reused illegally. However, 100% security can never be promised by any technology.

6.9 Social and Political

In this project there is a conflict between two standards. Information is always important for the entity not just for individual, but also for organizations and societies. Politically, preventing government from checking and controlling peoples' information is not feasible and this is a very looming security concern. The most important things to be considered are privacy of any person and security of society. This project inhibits any third person to involve in communication with others. This is logical and the best thing for privacy, if we look from the social point of view.

7 Conclusion

In this thesis a precise overview of Android mobile operating system, mobile payment gateways and Bitcoin systems is provided. First we implemented Cryptography Service Provider library to enhance security of mobile payment gateways, which provides easy to use API and hence reduces any developer's burden of security implementation. Furthermore, an instant messaging module with push notification system was also implemented. All secure connections adhere to concrete cryptographic standards. The developed solution provides strong authentication mechanism, confidentiality of exchanged messages with integrity, and protection by using digital signatures. Alongside the transaction, transparency of the implementation to end user is outlined. An evaluation of the implemented system in terms of engineering standards is also discussed. Additionally, the implication of implementation in the context of mobile applications, especially mobile financial sector is also articulated. All in all, anonymity of Bitcoin, one of the biggest problems so far, is left as a future work. Suggestions for this solution are using TOR which is in the early stages of adoption and development or using a new address for every transaction which is one of the officially encouraged ways to make attacks of disclosing identity of users more difficult.

References

- [1] J. Dankers, T. Garefalakis, R. Schaffelhofer and T. Wright, "Public key infrastructure in mobile systems," *Supported by European Commission through the IST Programme under Contract IST-2000-25350*, October 2002
- [2] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", "*Internet Request for Comments, vol. RFC 5280 (Proposed Standard)*", May 2008. Available at: <http://www.rfc-editor.org/rfc/rfc5280.txt>
- [3] "Certificate and Public Key Pinning", [Online]. Available at: https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- [4] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh and V. Shmatikov, "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software", "*CS'12*", 2012 Available at: http://www.cs.utexas.edu/~shmat/shmat_ccs12.pdf
- [5] T. Dierks, E. Rescorl, "The Transport Layer Security (TLS) Protocol Version 1.2", "*Internet Official Protocol Standards, vol. FRC 5246*", August 2008. Available at: <http://tools.ietf.org/html/rfc5246>
- [6] Jive Software Open Source community, "About Open-fire", [Online]. Available at: <http://www.igniterealtime.org/about/index.jsp>
- [7] Wikipedia, "Openfire", [Online]. Available at: <http://en.wikipedia.org/wiki/Openfire>
- [8] Jive Software Open Source community, "Openfire Projects: Smack API", [Online]. Available at: <https://www.igniterealtime.org/projects/smack/>
- [9] XMPP Standards Foundation, "XMPP About", [Online]. Available at: <http://xmpp.org/about-xmpp/>
- [10] Saint-Andre, P, "Extensible Messaging and Presence Protocol (XMPP): Core", "*IETF. RFC 6120*", [Online], May 4, 2014. Available at: <https://tools.ietf.org/html/rfc6120>
- [11] Jive Software Open Source community, "User Service Plugin Read Me", [Online]. Available at: <http://www.igniterealtime.org/projects/openfire/plugins/userservice/readme.html>
- [12] Brito Jerry, Castillo Andrea, "BITCOIN A Primer for Policymakers". Available at: http://mercatus.org/sites/default/files/Brito_BitcoinPrimer.pdf
- [13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system". Available at: <https://bitcoin.org/bitcoin.pdf>
- [14] Google Finance, "1 BTC to USD". Available at: <http://www.google.se/intl/en/googlefinance/disclaimer/>

- [15] Block Chain Info, "Total Bitcoins in circulation". Available at: <https://markets.blockchain.info/>
- [16] Franco Pedro, *Understanding Bitcoin*, Wiley Finance Series, 2015. ISBN:9781119019152
- [17] Android Developer,"Storage Options". Available at: <http://developer.android.com/guide/topics/data/data-storage.html>
- [18] OpenSSL, "About the OpenSSL Project". Available at: <https://www.openssl.org/about/>
- [19] BouncyCastle Organisation,"About BouncyCastle". Available at: <http://www.bouncycastle.org/releasesnotes.html>
- [20] BouncyCastle Organisation,"About SpongyCastle". Available at: <http://rtyley.github.io/spongycastle/>
- [21] Wikipedia, "Bouncy Castle", [Online].Available at: <en.wikipedia.org/wiki/BouncyCastle/>
- [22] RFC 4211, "PKCS#10: Certification Request Syntax Specification",RSA Security,2005.Available at: <https://www.ietf.org/rfc/rfc4211.txt>

Non-exclusive licence to reproduce thesis and make thesis public

I, Sevil GULER (date of birth: 28th of April 1989),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Secure Bitcoin Wallet

supervised by Sead Muftic and Vitaly Skachek

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu/Stockholm, 15.06.2015