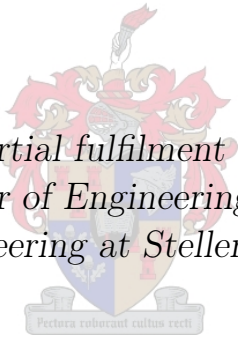
Bitcoin payment framework on a social media platform

by

Wessel Wessels



Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Prof. G. van Rooyen

December 2015

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2015/12/10 .....

Abstract

Bitcoin payment framework on a social media platform

W. Wessels

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (E&E)

December 2015

English abstract to be written

Uittreksel

Bitcoin betalingsraamwerk op 'n sosiale media platform

("Bitcoin payment framework on a social media platform")

W. Wessels

Departement Elektries en Elektroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (E&E)

Desember 2015

Afrikaanse uittreksel wat nog geskryf moet word

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations ...

Dedications

Hierdie tesis word opgedra aan ...

Contents

List of Figures

List of Tables

Nomenclature

No nomenclature yet.

Chapter 1

Introduction

Bitcoin [3] is a peer-to-peer decentralised digital payment mechanism that was introduced in a paper published by a person or group called Satoshi Nakamoto in 2008. Bitcoin aims to be an alternative to traditional centralised online payment mechanisms like credit cards and PayPal [4].

We focus on testing the viability of Bitcoin as payment mechanism on social media platforms, specifically on a mobile platfom.

To test the viability, we will design a Bitcoin payment framework that allows developers to incorporate Bitcoin payments in their applications without having to understand the low-level details of Bitcoin and without running any Bitcoin software.

We will also design and build a Bitcoin wallet application that will run on the social media platform to make payments directly from the application.

1.1 Background

Making payments on a mobile social media platform has been done in several different ways in the past. One of the biggest challenges to overcome is that many users don't have a credit card. For many social media platforms, this is especially true since the target audience of the platform is teenagers.

With Apple's in-app purchases [5], credit card details have to be loaded in only once. A parent can thus load a credit card for a child and let the child use it, but this can lead to problems of unregulated spending.

Another common method of payment is using mobile airtime as payment. Airtime is very easy to use, but the cost to the merchant is very high.

The last payment method we look at is a voucher based system, like Apple's Gift Cards. Vouchers are very similar to airtime as payment, but the voucher credit can only be used on the specific platform. Vouchers, like airtime, can be purchased at supermarkets. However, since the voucher is issued by the company with only the supermarket as middle-man, it follows that the loss that the company makes on overhead is less than with airtime.

Credit Card	Debit Card	PayPal
48%	30%	12%

Table 1.1: Table of preferred online method of payment [1]

We would like to test the viability of Bitcoin as an alternative to these methods of payment, especially to take advantage of Bitcoin's low transaction fees.

Bitcoin transactions can be processed without any transaction fee, but a small fee may be required for transactions that uses a lot of unspent outputs and thus causes a larger transaction. A large transaction in Bitcoin refers to the amount of bytes that the transaction uses and not the value of the Bitcoin. A small fee can also be added to speed up the processing time of the transaction. There are criticisms that say that the low price of transaction fees are unsustainable and will likely increase in the future [6].

1.2 Related Work

To compare Bitcoin as a payment method on a social media platform, we must look at existing payment methods and their advantages and disadvantages.

1.2.1 Credit and Debit Cards

According to a study done in 2014 [1], 48% of people prefer to make online payments with credit cards and 30% prefer debit cards. It is clear from these statistics that these traditional payment methods are still very prevalent.

1.2.1.1 Advantages

Credit card payments are easy to make and are processed quickly. The cost of transactions are very difficult to determine, as they differ from bank to bank, where you buy and the user's specific credit card package. However, credit card payments can be very cheap, since they usually have a small fixed and percentage fee per payment that is paid by the merchant. Since credit cards are backed by a trusted third-party, they are able to provide mediation when desputes arrive. They can also provide insurance against fraud.

1.2.1.2 Disadvantages

According to a survey done in 2014, 75% of adult South Africans are banked [7]. A quarter of the adults do not have bank accounts, making it almost impossible for them to purchase anything online. Furthermore, there are prerequisites to

SMS Cost	Vodacom	MTN	Cell C	8ta
R1.00	R0.07	R0.06	R0.09	R0.04
R1.50	R0.44	R0.31	R0.37	R0.27
R2.00	R0.81	R0.56	R0.66	R0.52
R3.00	R1.55	R1.08	R1.24	R0.99
R5.00	R3.03	R2.11	R2.40	R2.06
R7.50	R4.88	R3.39	R3.84	R3.31
R10.00	R6.73	R5.10	R5.29	R4.57
R15.00	R10.43	R7.24	R8.18	R7.07
R20.00	R14.13	R9.81	R11.14	R9.58
R25.00	R17.83	R10.91	R13.96	R12.10
R30.00	R21.53	R14.40	R16.85	R14.60

Table 1.2: Table of SMS payment payouts from bulksms.com [2]

getting a credit card, like minimum income. Credit cards also usually have a fixed monthly fee.

Using a credit card is not anonymous, since the user has to enter the card number for each payment. Every payment made by the user is thus traceable by the credit card company. This may be a concern to certain people.

Even though credit cards have very small transaction fees, they still have a small fixed fee. This is an issue when payments are very small. For example, the company PayFast [8] charges a fixed R2 and 3.9% of the payment as a fee. Thus, when a user wants to make a R5 payment, the merchant will only receive $R5 - R2 - R5 \times 3.9\% = R2.81$.

1.2.2 Airtime Payment

A popular mobile messaging application, Mxit, uses airtime payments with its Mxit Moola [9] virtual currency. They use a system where a user sends an SMS to a specific number to make a payment. For example with Mxit Moola, a user will send an SMS costing R3 of their airtime and receive 300 Moola. This is very convenient for the user, however it is not very profitable for the company.

From table 1.2 and figure 1.1 we can see that the company itself gets a very small percentage of the payment. This is especially true for payments less than R5, where the average payout percentage is less than 50%. Even the highest SMS value at the network with the highest payout only has a payout of 71.77%.

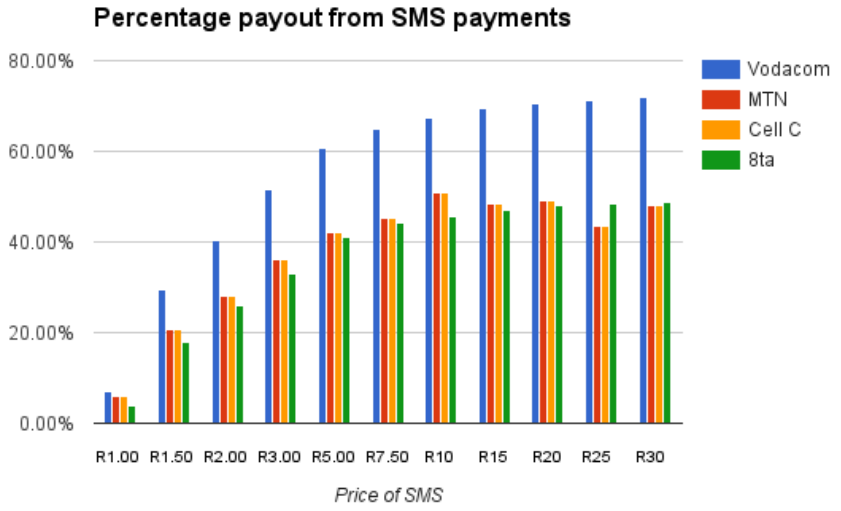


Figure 1.1: Graph showing payout percentages for SMS payments

1.2.2.1 Advantages of Airtime

The airtime model works very well, because it allows teenagers to use it with their existing airtime quotas without requiring their parents credit card details. Airtime is also very easily accessible, since it can be bought with cash at most supermarkets among other methods.

1.2.2.2 Disdvantages of Airtime

Airtime as payment only has one big disadvantage - the very high transaction fee paid by the merchant.

1.2.3 Vouchers

Vouchers are very similar to airtime in regards to accessability. A physical voucher can usually be bought at a supermarket, where a code is sold to the user. This code is then entered by the user on the service provided by the company that provides the voucher, where the user will then receive the amount of the voucher as credit. For mobile purchases, Apple's Gift Cards are a very good example of vouchers in practice. The gift cards can be used in all the same Apple-related online stores that credit cards are used. This allows users to effectively buy online virtual goods using cash.

1.2.3.1 Advantages of Vouchers

Vouchers are easily accessible and are purchasable using cash. They are comparable to the airtime payment model, but they provide more controll to the merchant. Vouchers also enable users to budget their spending, something that is ideal for teenagers and children.

1.2.3.2 Disadvantages of Vouchers

Vouchers can only be used at the online merchant that they are bought for. Thus, if a person wants to pay at different online merchants, they will need to buy a voucher for each of the merchants.

Since a voucher is a physical item, it has to be manufactured and distributed across the world. This adds to the overhead cost, and it is possible for vouchers to be sold out in certain places.

1.3 Objectives

Our objective is to test the viability of Bitcoin as a payment mechanism on a social media platform. The platform we chose is a text-messaging social media application called WeChat.

We want to test how Bitcoin compares to alternative payment mechanisms on the following criteria:

- Transaction fees
- Ease of use
- Versatility

Chapter 2

Literature Study

2.1 Bitcoin

Bitcoin is a highly technical protocol that was introduced in 2008. For this project we will look at the relevant practical characteristics of the Bitcoin network as it is implemented in the real world.

The code that is responsible for the Bitcoin network is open source and is maintained by a core team of developers. Since the code is open source, anyone can verify the code to ensure it is not malicious. The Bitcoin network also chooses what code to run, so consensus must be reached between the Bitcoin network and the software.

Bitcoin is decentralised, which means no central authority controls the network. Rather, Bitcoin is run by independant nodes that work together.

Bitcoin uses cryptography to remove trust from a central authority.

2.1.1 A Bitcoin Address

A Bitcoin address has two parts, a private key and a public key. The private key is an unsigned 256 bit integer that is usually randomly generated, but can also be chosen by the user. This private key should be kept secret, as anyone with the key can spend any Bitcoin that belongs to it.

The public key is generated from the private key using a combination of Elliptic Curve Digital Signature Algorithms and several hashing functions. A private key can't be determined by knowing the public key, and thus it is safe for the public key to be known by anyone.

An analogy of a postal box can be used to explain a Bitcoin address. The public key is like a post box. Anyone can deposit something into the box without having access to the key of the box. They only need to know the box number to make the deposit. To retrieve or spend whatever is in the box, we need the key. Thus, the private key is like the key to the post box. Only the person with the key can open the box. One important difference to a normal

postal box is that the postal box is completely transparent, meaning anyone can see exactly what is in the box.

2.1.2 How Bitcoins Are Stored

Bitcoin is "stored" using a public ledger called the blockchain. Effectively, every single successful Bitcoin transaction is stored in this public ledger, and every user that runs a full Bitcoin node has an identical copy of the blockchain. The authenticity of the blockchain is managed by using consensus on the network and using hashing alogrithms. Bitcoin transactions are bundled into blocks, and the entire block is hashed. This block is hashed with the hash of the previous block as input, and so forth in the chain until the first block is reached. Thus, every block's hash is the hash of the entire chain behind it. This means that nothing can be changed in the chain, since it will change the entire hash after the change.

Since the entire ledger of Bitcoin transactions are available, it is possible to calculate the balance of any address at any given time. Since there are so many copies of the blockchain, only the Bitcoin private key is required to be stored. Even though the public key is calculated from the private key, a server usually stores the public key as well since it will be inefficient to calculate it everytime we a lookup is needed. The fact that we only need to store these two values in order to have access to the Bitcoin is the most relevant part of how Bitcoin works for our purposes. It allows a developer to simply store these simple two values securely, without having to keep track of ledgers locally.

2.1.3 How a Bitcoin Transaction Works

Since the blockchain is a public ledger of all transactions, it is a crucial part of making a new transaction. Every transaction has inputs and outputs. When looking at an address at a specific point in time, we can determine which of the outputs of its transactions are not spent yet. They are called unspent outputs. These outputs can be cryptographically verified to belong to a specific address. These outputs can be used as inputs in a new transaction. The transaction then specifies new outputs to send Bitcoin to. To receive "change" in a Bitcoin transaction, the change amount must be specified to the address that the payer chose. The change address can be a new address owned buy the payer, or the same address that the payment came from.

The difference between the inputs and outputs of a transaction is the transaction fee that is claimed by users that verify transactions.

2.1.4 Bitcoin Mining

Bitcoin Mining is the process of bundling transactions together to form a block. As mentioned in 2.1.2, every block has a hash that is in effect the hash of the

entire chain. For a block to be valid, this hash has to satisfy the condition of leading with a certain amount of zeroes. This concept is arbitrary. Its only purpose is to be difficult to do, so that a lot of work is required to do so. The only way of doing this is using a brute force method. Thus, more computation power increases the odds of finding a block that satisfies the zeroes criteria.

The amount of zeroes required is called the "difficulty", since more zeroes make it more difficult to find a valid block. The amount of zeroes required is dynamically updated by the Bitcoin network to ensure that the average time for finding such a block is 10 minutes.

When a transaction is mined into a block, we say it has 1 confirmation. The longer the chain becomes after this block, the more confirmations it has and we can with higher trust say that the transaction is final. For example, a transaction with only one confirmation can still be rejected if a successful double-spend attack is done. With more confirmations, the probability of doing a double-spend attack lowers exponentially [3]. With very small payments, we do not require many confirmations to accept a payment. It is worth the risk of accepting a payment with 0 confirmations, since it is not worth the effort to try and do a double-spend on such a small transaction.

Chapter 3

System Design

3.1 Framework

In this project we test the viability of a payment framework on a mobile social media platform. The framework will consist of several independant but connected pieces:

- REST API for payments
- Wallet Application
- Bitcoin Interface
- Use-case Application

A summary of what is required from the system can be seen in figure 3.1.

3.1.1 REST API for payment management

A REST (Representational State Transfer) API [10] was chosen for the main interface for developers to use Bitcoin without running a Bitcoin node or having experience with Bitcoin. REST was chosen because it is a commonly used architecture, it is easy to use and understand and it does not constrain the user's choice of programming langauge or environment.

The purpose of the REST API is to let developers make payment requests and check if a payment has been made, without dealing with the low-level Bitcoin transactions directly. Thus, our system should generate a new Bitcoin address on request.

Our requirements from the REST API are:

- New Bitcoin address for each payment
- Verify payment

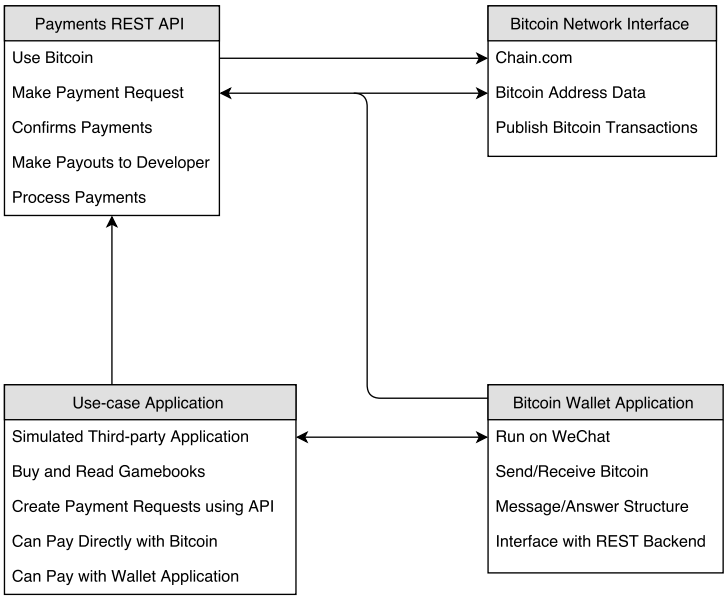


Figure 3.1: Summary of Framework

- Check total balance of developer
- Witdraw available Bitcoin of developer

3.1.1.1 The concept of the Bitcoin payment

This is a high-level explanation of how to receive verifyable payments with Bitcoin. With Bitcoin, unlike a traditional bank account, you don't have a single "account" where people can make payments to and you can verify that the payment came from them. With Bitcoin it is trivially easy to make a new Bitcoin address, and it can be generated without being connected to the Internet or the Bitcoin network.

Since the entire Bitcoin blockchain is public, a single address is not sufficient to receive multiple payments. With a single address, it is not easy to verify that a spesific person has made a payment, since there may be several payments of the same amount happening in short succession.

The sollution to the problem is generating a new address for every payment, and requesting that the user make the payment to that address. Since the newly generated address is not yet present on the blockchain, when a payment to that address of the requested amount occurs, we can be certain that the person in question made the payment. When the payment is complete, the Bitcoin in that address can be transferred to a central address, and the original address can be discarded.

From our requirements for the REST API, we clearly require (at least) the following methods:

- A payment method
- A balance method
- A payout method

3.1.1.2 The /payment method

The /payment method is the core of the REST API. It is used to make a payment request with a specified amount of Bitcoin and a description of the transaction. The /payment method returns a Bitcoin public address and a payment ID.

The user can then pay to the Bitcoin address using any standard Bitcoin payment method, or can pay directly from the Bitcoin wallet that will run on WeChat and will be connected to the payment infrastructure.

3.1.1.3 /payment/{PUBLIC_ADDRESS} and /payment/{ID}

These two methods are conceptually the same, but they take in two different arguments. The one takes the Bitcoin address to be queried, and the other takes the payment ID. The method returns all the data about the transaction, including the status of the transaction.

The main purpose of this method is to verify that a transaction has been completed by the user. It can also be used to give the payment details to to user again.

3.1.1.4 /balance

The /balance method gives the developer the balance of all the available Bitcoin from all the received transactions. The method also returns a flag that says if there is enough Bitcoin to make a payout.

3.1.1.5 /payout

The /payout method is used by the developer to transfer all of the available Bitcoin to a specified Bitcoin address.

3.1.2 Wallet Application

The Wallet Application is a Bitcoin wallet implemented on the WeChat platform. The WeChat platform uses a simple message-answer structure. A user sends a message in the Wallet Application. The message is then sent to WeChat that sends it to a third party server controlled by the developer. The server then sends a reply to WeChat that is then forwarded to the user.

In this manner, a fully functional Bitcoin wallet is realised. The third party server stores the private keys and processes the Bitcoin transactions on commands from the user.

The advantage of using the WeChat platform is the security built in to the platform, as well as an existing userbase.

The Wallet Application will be directly connected to the back-end of the REST API. Thus, payment requests will be referable directly from the Wallet Application without needing to reference the Bitcoin Address. It will be able to reference the request using the payment ID mentioned in 3.1.1.2

3.1.3 Bitcoin Interface

To connect to the Bitcoin peer-to-peer network, a Bitcoin client is needed. The standard way of doing this is running the Bitcoin open source software on a server. This is very network and processor intensive. For development and testing, it will be quite expensive to run the Bitcoin software. Thus, an alternative is required for interfacing with the Bitcoin network. Fortunately, there are services that provide access to most of the Bitcoin operations using their API's.

We require the following from such a service:

- Get the balance from an address,
- Get unspent outputs from an address,
- Post a signed Bitcoin transaction to the network,
- It must be able to use the Testnet

The details of the chosen service is covered in chapter 4.

3.1.4 Use-case Application

To use the payment framework, a Gamebook application is created to read Choose Your Own Adventure style books on the WeChat platform. User-created books can be sold by using the Bitcoin payment framework and the author can potentially earn Bitcoin.

For each sale, the Gamebook application creates a payment request using the REST API. The user can then pay using the Wallet Application or any Bitcoin payment mechanism. The Gamebook application can the query the API to confirm that the payment is received.

Chapter 4

Detail Design

4.1 Back-end

To implement the payment framework, we need a back-end server and a back-end web framework.

4.1.1 Back-end Server

We chose an Amazon Elastic Cloud Computing (EC2) instance for the back-end server. EC2 gives us access to a virtual machine (VM) where we can run our own software, including a publicly accessible website. The micro instance is deployed in Singapore in the Asia Pacific region. The reason for this is to have the server as close as possible to the WeChat servers in China, since our servers must connect to WeChat's servers.

4.1.2 Back-end Web Framework

We decided to use XAMPP (Apache, MySQL, PHP and Perl) for the back-end development environment. XAMPP is a full stack development environment. It includes an HTTP server (Apache), a database server (MySQL) and a scripting language (PHP). XAMPP is free and easy to deploy, and is also used because the author is familiar with it.

4.2 Social Media Platform

We chose WeChat for our social media platform. WeChat works on most smartphones, already has a userbase and it has a third-party API with a development sandbox feature.

On WeChat, a third-party application is known as an "Official Account". We registered for a sandbox Official Account that only allows 20 users and is not searchable on WeChat.

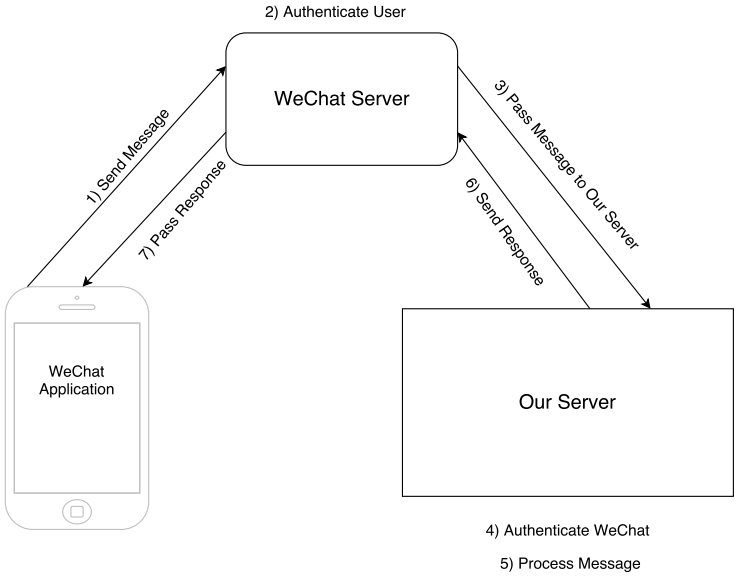


Figure 4.1: Interaction with WeChat

WeChat acts as an intermediary between the user and our server as seen in figure 4.1.

4.2.1 WeChat Security

WeChat uses a shared token hashing scheme to authenticate itself on our server. We provide WeChat with a unique string to use for authentication purposes. That string is then used in every request that is made to our server. WeChat uses the string we provided, a random nonce and the UNIX timestamp to create a signature. It sorts the string, timestamp and nonce and forms a single string from these three values. This single string is then hashed using the SHA256 hashing algorithm to generate a signature. When a request is made to our server, WeChat provides the nonce, the timestamp and the signature The message does not contain the unique string. Since we know the unique string, we can use the nonce and timestamp to also generate a signature. If the message comes from someone that possesses the same unique string, the signature that is provided must match the signature that we calculated.

Thus, we can be reasonably certain that any request that is made to our server with a signature that is verified comes from WeChat and not an attacker. If our unique string is compromised somehow, someone will be able to make false requests to our server that appear valid. If this would happen, we can give WeChat a new unique string.

4.2.2 WeChat Interface

The WeChat Official Account interface works on a message-answer basis. Any message that the user sends in the Official Account dialog is forwarded to our server. When we configure our Official Account, we provide WeChat with a URL that points to the script that handles all messages from and to Wechat. This script verifies any incoming messages as described in section 4.2.1.

The incoming message is in XML format. It contains a unique identifier for every subscriber to the WeChat Official Account. It is important to note that the unique identifier is only unique for the spesific Official Account. The identifier is not a global unique identifier for the entire WeChat platform. Thus, the same user will have two different identifiers in two different Official Accounts. This is important to remember, since we will be using two seperate Official Accounts.

The messages that are received must be interpreted and responded to accordingly. Since we will have applications that rely on previous messages and results, we need something to keep track of the user's current state. A state machine is required to look at the current state the user is in, look at the message they sent and accordingly determine what to reply and to what state to move to.

To keep track of the user's state, we will use a MySQL database as described in section 4.1.2. Thus, for each message received, we will read the user's current state from the database and apply the state machine logic to their message. We will then update their state in the database and reply with the message that the state machine determined.

4.2.3 Gamebook Design

The use-case application we chose to demonstrate the payments framework is a Gamebook application. A Gamebook is a non-linear book that tells a story based on the user's input. In physical books, this is done by giving the user a choice and then telling them what page to go to based on their choice. In the digital realm, we can simply provide the user a choice and give them the corresponding text. The software keeps track of what options links where.

In a dedicated Gamebook reader, a common method of generating a Gamebook is using a scripting language like ChoiceScript [11]. Since WeChat doesn't have any client-side scripting, using a scripting language like this is not possible.

We decided to create our own system of storing and reading Gamebooks. We use the MySQL database as the method of storing and organising the Gamebooks.

ID	Text	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5

Figure 4.2: Gamebook Database Table

4.2.3.1 Gamebook Database Design

Every book in our design is a table, and every page is an entry in the table. Each entry has a unique id, the text of the page and then the references of each of the choices from that page. We chose five to be the maximum amount of choices on each page. The database resembles a linked list, where each entry has pointers to the next corresponding entry. The design of the Gamebook table is seen in figure 4.2.

The values of the fields choice 1 - 5 will have the corresponding ID's of the "page" they link to. If an entry only has two choices, the rest of the options will have the value null to indicate that it is not a valid choice.

We chose this method of storing the Gamebooks, because it is easy to implement, easy for a writer to visualise and allows us to have sircular stories and also allows us to visit multiple branches of the story.

We will also need two more tables: one to store a list of all the books available and another to keep track of books purchased by individuals.

4.2.3.2 Gamebook Author Platform

As stated in section 4.2.3, we are not going to use a scripting language to write the Gamebook in. It doesn't fit the design of the application that we are building, and has a learning curve for people that want to start writing books and are not familiar with scripting languages. We decided to use a Graphical User Interface for writing the Gamebooks.

We decided to use a website as the author platform. We used commonly used web technologies to build the platform. For the front-end, we used HTML, JavaScript with jQuery, and Twitter's Bootstrap CSS framework.

Using AJAX, we connect the front-end with the back-end PHP scripts that connects to the MySQL database. We also decided to use an OAuth login mechanism to facilitate logins, rather than writing our own login system. A big motivation for doing this is the authors desire to get experience with OAuth. We decided to use Google's OAuth platform, because it widely used and well documented. By using OAuth, we are enabled to have unique, secure logins without having to design all the security measures to protect the user. It also makes it easier for the user, because they can log in with a single click.

Since the Gamebook resembles a tree, we decided to display the tree using an organisational chart. We used Google's JavaScript Chart API to map the Gamebook. The Chart API is free, simple, powerful and well documented. To display the story as a tree makes it easier for the writer to navigate through the book and to get a bigger picture of the story. An example of such a chart

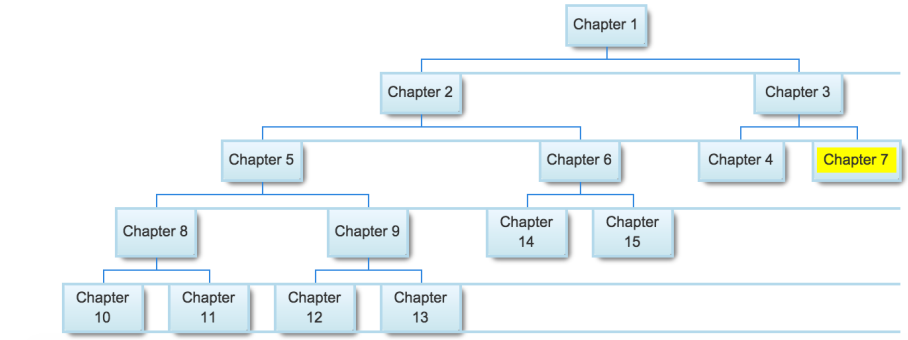


Figure 4.3: Gamebook Tree Representation

can be seen if figure 4.3. Using a chart makes it easier for the author to see where the story needs more work.

4.3 WeChat Wallet Design

As stated before, the WeChat Official Account doesn't allow us to run client-side code. This means that the WeChat Wallet will have to be a hosted Bitcoin wallet, with the Official Account interfacing with the hosted wallet.

To have a WeChat wallet, we need a Bitcoin address. When hosting a Bitcoin wallet, there are two main method of keeping addresses. The first method is where a user gets a single address or adresses and these addresses are used only by the single user. The user may or may not have access to the private key, but the wallet is associated with only that user. This allows the user to verify the balance and transactions by checking the blockchain.

The second method is where the user has an account, and can have a receiving address, but the user is not directly in control of the address. The users' balance is not verifiable by using the Bitcoin blockchain. This is because the server keeps track of the balance of the user, and makes a payment from other addresses when a user wants to make a payment.

We chose the first method of storing the address, because it is simpler and has a more direct feeling for the user that he is using a real Bitcoin wallet, and not just something that arbitrarily keeps track of the balance.

4.3.1 Bitcoin Interface Design

Thus, we need to generate a Bitcoin address for each user. To do this, we use a a popular open-source PHP Bitcoin library called bitcoin-lib-php. One of the features of Bitcoin is that generating a Bitcoin wallet can be done locally. That means that the bitcoin-lib-php library generates a Bitcoin address on our server without connecting to the Bitcoin network. This has many advantages,

including it doesn't use network bandwidth, it is fast and the private key never leaves our server.

We chose bitcoin-lib-php as our library, because it satisfies our requirements perfectly, it is open-source (and thus verifiable) and is decently documented.

As mentioned in section 4.3, by creating an address for each user, the user can more directly monitor his funds and transactions since it can be independently verified by using any software that is connected to the Bitcoin blockchain.

The Bitcoin library allows us to create an address, create a Bitcoin transaction and sign a transaction. These are the core functions of Bitcoin. However, a transaction can't be created without information about the address or addresses that want to create the transaction. This information is called "unspent outputs" and, as the name suggests, are previously received transactions that are not yet spent. These unspent outputs contain all the information to build a transaction, including the value of the output, the transaction hash and the script type etc.

These unspent outputs are only acquirable with access to the Blockchain and we do not run the Bitcoin software. Thus we do not have direct access to these outputs. From our requirements in section 3.1.3, we require a third-party Bitcoin interface to provide these outputs among other things. From the requirements listed in section 3.1.3, we decided to use chain.com. Chain.com satisfies all our requirements and is free at the time of writing.

During the progress of this project, chain.com has release version 2 of their API with the biggest change being that they can also sign a transaction for you. When we started, we had to sign the transaction ourselves. Signing the transaction ourselves exposes us to a smaller risk of compromising the security of the users' private key. We choose to still use version 1 of the API, because updating to version 2 will not have a significant effect on our proof of concept's overall outcome. The summary of the Bitcoin payment flow can be seen in figure 4.4.

4.3.2 WeChat Wallet Features Design

The main function of a Bitcoin wallet application is to send and receive Bitcoin. A Bitcoin address is a string of 26 to 35 alphanumeric characters. Reading an address to someone is not practical due to the long length. A very common method of representing a Bitcoin address is by using a graphical QR-code. Another user can then scan a QR-code and receive the address to make the payment to.

4.3.2.1 Address to QR-code

Our first design specification is to represent the user's alphanumeric address as a QR-code image. There are several ways we can do this. The first method

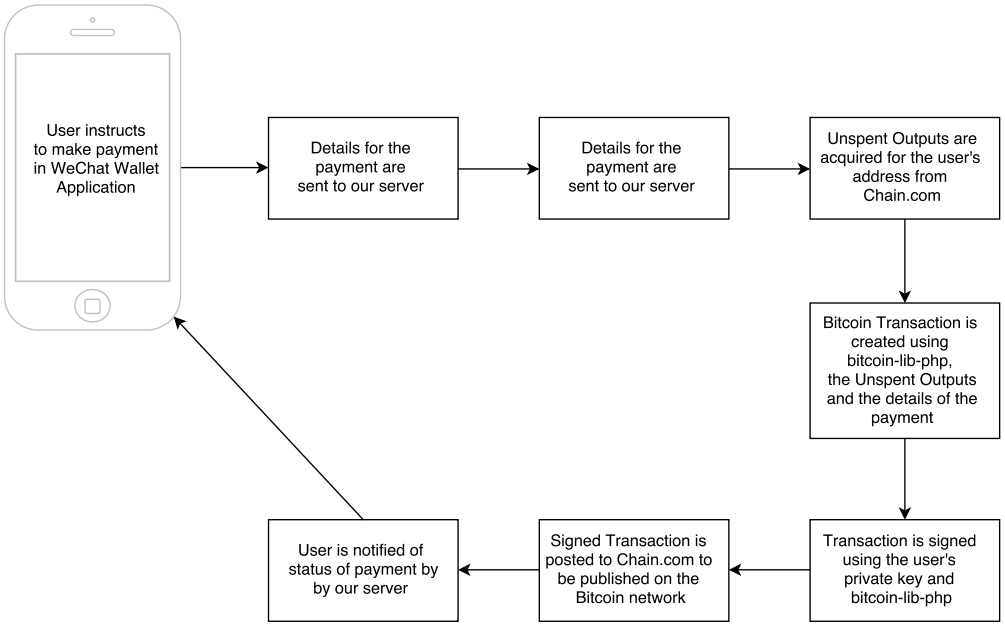


Figure 4.4: Bitcoin Payment Flow

is to generate the QR-image on our server and send the image to the user on WeChat. The problem with this method is that our server will be responsible for generating images, and this is computationally intensive. The second method is using a webpage that generates the QR-code client-side using JavaScript. The WeChat Official Account allows us to open a link directly in WeChat. Thus, we can send the user a link of a webpage that will generate the QR-code in WeChat's built-in browser. With this method, computation is spread to the users and not a central server.

We chose the webpage method of generating the QR-codes. We wrote our own webpage that uses an Open Source JavaScript library to generate the QR-code. We encode the address to be generated in a GET request and send the link to the user. The link will look like this:
https://domain.com/qr?address=1Bd5wrFxHYRkk4UCFttcPNMYzqJnQKfXUE
This QR-code can then be generated when a user wants to receive a payment from someone. An example of a generated QR-code can be seen in figure 4.5.

4.3.2.2 QR-code to Address

Our second design spesifiction is to allow a user to scan a QR-code to make a payment to. Although the WeChat application has a QR-code reader built-in, it doesn't allow the user to scan a QR-code directly from an Official Account. An Official Account does allow a user to send an image to our server. This will allow us to decode the QR-code on our server. For simplicity, we decided to use a third-party service called goQR.me to decode the QR-code. WeChat

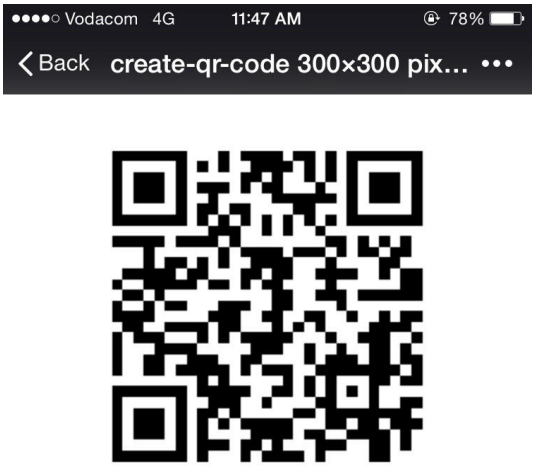


Figure 4.5: Generated QR-code

sends our server a link of the image that the user sent. This link of the image is used by the third-party service and responds with the decoded text. The bitcoin-lib-php software can verify that the decoded text is a valid Bitcoin address.

This method of decoding the QR-code is very computationally- and bandwidth efficient for our server, since the image itself is not sent through our server. Just like the Bitcoin Interface in section 4.3.1, we chose to use a third-party service for our proof of concept. If we were to build the service for production scale, we would rather exchange the third-party service for our own software. In the case of the QR-code service, it would be a major security concern to use a third-party service. A malicious third-party can exchange the correct address with their own address to receive Bitcoin that is not intended for them.

4.3.2.3 User Balance

Getting the balance of the user's address is the second core feature that a Bitcoin wallet application requires. Calculating an address' balance involves summing all the user's unspent outputs. Fortunately, the Chain.com API allows us to retrieve the balance of any Bitcoin address directly using a single method call. The result of getting the balance can be seen in figure 4.6.

4.3.2.4 Make A Payment

The core process of making a payment is handled in section 4.3.1 and can be seen in figure 4.4. The most important part of the payment flow is that the user's Bitcoin private key never leaves our server, even though a third-party service is used to publish the transaction.

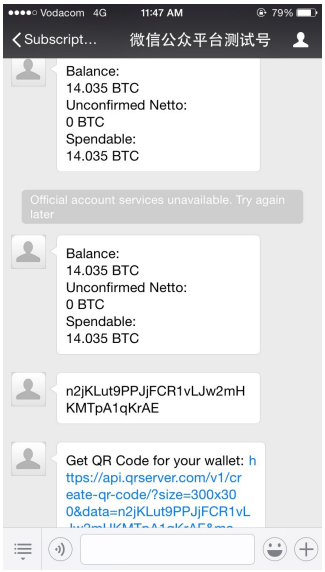


Figure 4.6: WeChat Balance

4.4 REST API Design

There are many ways to implement a REST API. This is because of the "black-box" approach of REST. As such, the first major design decision that needs to be made is whether to use a REST framework or two build it from scratch. Using a REST framework would be easier and standardized and if the framework is maintained it would also be secure.

In spite of the advantages of using a REST framework, we decided to build it ourself. The motivation for choosing this method is that the author wanted to get a better understanding of how a REST API works on the low-level and felt that building it from scratch would give a better understanding. Thus, we will use the same tools to build the other parts of our system, ie. PHP and MySQL.

4.4.1 REST Endpoints

A PHP script is usually called in the following format: "url/script.php". However, for a REST API we need a clean URL in the format: "url/script". Since we are using an Apache server, we need to modify the ".htaccess" file with regular expressions.

Thus, we have different regular expressions in the .htaccess file that will call a single script with the corresponding parameters from the request. The .htaccess file will only relay the parameters of the url to the PHP script. The PHP script will then apply logic to determine what function should be executed.

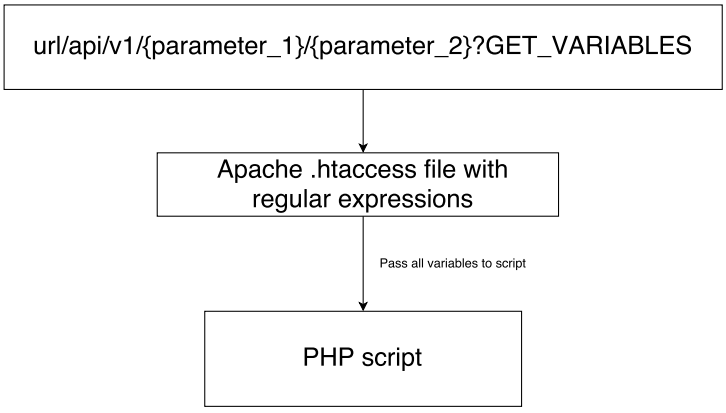


Figure 4.7: REST URL Flow

Flag	Description
NC	No Case. Doesn't differentiate between upper- and lower case.
QSA	Query String Append. Adds the variables to the existing GET variables.
L	Last. If the rule matches, no further rules will be processed.

Table 4.1: Table of selected .htaccess flags

The PHP script will firstly authenticate the developer, followed by the do the corresponding external API calls, database entries, response HTTP status code and JSON response data.

The following is an excerpt from the .htaccess file that is responsible for the routing of the REST endpoints:

```
RewriteRule ^api/v1/([A-Za-z0-9-]+)/([A-Za-z0-9-]+)/?$
/chaintest/my_api.php?first_query=$1&second_query=$2 [NC,QSA,L]

RewriteRule ^api/v1/([A-Za-z0-9-]+)/?$ /chaintest/my_api.php?first_query=$1 [NC,QSA,L]

RewriteRule ^api/v1/?$ /chaintest/my_api.php
```

The first RewriteRule matches queries with two parameters in the URI and turns the parameters into GET variables first_query and second_query. The my_api.php script will then be able to access these variables like normal GET variables.

Chapter 5

Tests

5.1 Quantitative

This project is an implementation of a platform on top of an existing social media platform WeChat. We do not have access to the back-end systems of WeChat. This makes it hard to do many of the quantitative tests required for the application itself. They will require qualitative tests. The REST API however, can be tested thoroughly using unit tests.

5.1.1 REST API unit tests

There are tools and frameworks available for testing REST API's. Just like when making the REST API, we decided to rather write the unit tests ourselves without using a framework in order to get a better understanding of how unit tests work.

Thus, we wrote the entire REST API unit tests in a PHP script that calls all our REST endpoints with inputs that we know what the outputs should be. Each test should also confirm that the correct HTTP status code, as shown in table 5.1, is returned. In total, there are 39 different tests that excecute.

Status Code	Text	Description
200	OK	Request succeeded
400	Bad Request	Request not understood due to malformed syntax
401	Unauthorized	The request requires user authentication
404	Not Found	Request does not exist

Table 5.1: Table of selected HTTP status codes

5.1.1.1 Testing Authentication

In order to test the authentication of the developer with our REST API, we wrote a simple endpoint called /test that returns an HTTP status code "200 OK" if the authentication was successful and the corresponding status code if it is unsuccessful.

The following are the tests that should not authenticate:

- Duplicate Request
- Missing api_key
- Invalid api_key
- Missing nonce
- Missing timestamp
- Missing signature
- Invalid signature

Since all of these tests fail to authenticate, they should all return an HTTP status code "401 Unauthorized".

All further tests assume that the authentication happens successfully.

5.1.1.2 Testing /payment

To test the /payment endpoint, we make a payment request with an amount and a description. For the result, we expect a valid Bitcoin address with the same amount and description that we provided. We uses a function in bitcoin-lib-php library to validate the Bitcoin address.

The following are tests that should fail:

- Wrong Method (GET instead of POST)
- amount_sat smaller or equal to 0
- amount_sat not an integer
- Missing description
- Description too long

These tests should all return an HTTP status code "400 Bad Request".

5.1.1.3 Testing /payment/{TRANSACTION_ID}

To test this method, we need to call it with a valid transaction ID that corresponds to the api_key that created the transaction. If the ID does not correspond with the api_key, the test should return an HTTP status code "401 Unauthorized".

5.1.1.4 Testing /payment/{PUBLIC_ADDRESS}

Testing this method is very similiar to the /payment/{TRANSACTION_ID}. The public address must correspond with the api_key that created the transaction.

The following tests should fail:

- Wrong Public Address (401 Unauthorized)
- Invalid Public Address (400 Bad Request)

5.1.1.5 Testing /balance

This method only has no arguments. Thus, if the authentication succeeds, it should return an HTTP status code "200 OK" and the balance.

5.1.1.6 Testing /payout

This method will always return an HTTP status code "200 OK" with the result in the JSON response.

The following tests should fail:

- No Address Given
- Invalid Address

5.1.1.7 Testing Invalid API Method

All these tests should return an HTTP status code "404 Not Found":

- No Method Given
- Testing "/"
- Testing /asdfasdf
- Testing /asdfasdf/asdfasdf
- Testing /balance/asdfasdf
- Testing /payment/asdfasdf

5.2 Qualitative

Since we are implementing a system that is used by people, a large part of our testing is qualitative.

We wrote a guide that goes step-by-step through the processes of using the systems that we have built.

The user is asked to add the Wallet Application inside WeChat and then load some Testnet Bitcoin to their wallet from a website that gives free Testnet away. This is to let them understand the concept of loading Bitcoin to their Wallet. They are then asked to write a small Gamebook on the Gamebook Author page. They only need to write three pages: one start page that links to two pages.

After they finished the Gamebook, they are asked to add the Gamebooks Application inside WeChat. They should then navigate in the Gamebooks Application to the small story they just wrote and buy it. When they select it, they will receive a payment ID that they must enter in the Wallet Application and confirm the payment. They should then be able to read the story they just wrote and bought.

To get feedback, we wrote a Google Form to make a questionnaire. The advantage of using a Google Form is that it directly outputs the answers of the users in a single spreadsheet.

The main things we wanted to learn from the feedback was how easy the users where able to use the payment mechanism and how viable it is compared to traditional payment mechanisms.

5.2.1 Feedback

90% of users used an iPhone when testing. The reason for this being so high is that a phone was offered for them to use in case they didn't have WeChat installed on their own phone. 10% of users had an Android device, and the test was successful.

60% of users never used WeChat before this test, and the other 40% have used it before, but they no longer use it.

20% of users are familiar with Bitcoin, with 70% saying they have heard about it and 10% saying they have never heard about it.

90% of users where able to add the Wallet and Gamebooks Applications within WeChat on the first try, with 10% being able to add it with some trouble.

60% of users said that the Wallet Application is easier to use than traditional payment mechanisms, with 20% saying they are the same an 10% said that traditional payment mechanisms are easier.

50% of users said that the Wallet Application is faster than traditional payment mechanisms, with 30% of users saying they are the same and 20% saying that traditional mechanisms are faster.

	Average	Mode	Std. Dev.
Wallet	7.8	7	1.48
Gamebooks App	8	8	1.33
Author Page	8.4	9	0.97

Table 5.2: Table of ratings of individual parts of the system

We asked users what payment mechanism they would prefer to use assuming that Bitcoin is more easily accessible. 50% of users said they would prefer Paypal, with 40% of users saying they prefer the Wallet Application and 10% saying they prefer a debit card.

The users were asked to rate the Wallet Application overall out of 10. The average rating is 7.8, with the mode being 7 and a standard deviation of 1.48.

For the Gamebooks Application, we only look at a rough experience. All of the users were able to write a Gamebook on the author website, with the majority saying it was easy to write. All of the users were able to select the book they wrote on the app, purchase it and read it.

The users were also asked to rate the Gamebooks Application overall out of 10. The average rating is 8, with the mode also 8 and a standard deviation of 1.33.

For the Gamebooks Author page, the average rating is 8.4, with the mode being 9 and the standard deviation being 0.97.

5.2.2 Feedback Interpretation

The response to the applications that was tested was generally positive. The majority of the users quickly grasped the concept of making a payment from the Wallet Application to a 3'rd party application.

Chapter 6

Conclusion

6.1 The Conclusion

Appendices

Appendix A

No appendix yet

Bibliography

[1] I. Total System Services, "2014 Consumer Payments Study," Tech. Rep., 2014. [Online]. Available: http://tsys.com/2014ResearchReport/

[2] Bulksms.com, "Premium Rated SMS." [Online]. Available: http://www.bulksms.com/products/premium-rated-sms.htm

[3] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Consulted*, pp. 1–9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[4] PayPal, "PayPal," 2015. [Online]. Available: https://www.paypal.com/za/webapps/mpp/about

[5] Apple, "Make in-app purchases." [Online]. Available: https://support.apple.com/en-za/HT202023

[6] K. Kaskaloglu, "Near Zero Bitcoin Transaction Fees Cannot Last Forever," in *The International Conference on Digital Security and Forensics (DigitalSec2014)*, 2014, pp. 91–99.

[7] FinMark Trust, "FinScope South Africa 2014," Tech. Rep., 2014.

[8] PayFast, "PayFast Fees." [Online]. Available: https://www.payfast.co.za/fees/

[9] Mxit, "What is Moola." [Online]. Available: http://web.support.mxit.com/articles/moola/what-is-moola

[10] Oracle.com, "What Are RESTful Web Services?" [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

[11] C. o. G. LLC, "Introduction to ChoiceScript." [Online]. Available: https://www.choiceofgames.com/make-your-own-games/choicescript-intro/