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-J 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

Acronyms and Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
EC2	Elastic Cloud Computing
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
SHA	Secure Hash Algorithm
SMS	Short Message Service
SQL	Structured Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
QR Code	Quick Response Code
XAMPP	Cross-platform Apache, MySQL, PHP and Perl
XML	Extenible Markup Language

Chapter 1

Introduction

Bitcoin [2] 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 centralised online payment mechanisms like credit cards and PayPal [9].

This thesis focuses on testing the viability of Bitcoin as payment mechanism on social media platforms, specifically on a mobile platfom. We want to investigate the practical viability of using Bitcoin to make small payments at a small or zero fee. One of our main goals is to compare Bitcoin with current payment mechanisms in terms of transaction fees. Bitcoin fees are not mandatory in all cases. When fees are required, the value of the fee is not fixed [10]. We want to investigate how these fees work and practically confirm that we can make payments with very small transaction fees.

To test the viability, we will design a Bitcoin payment framework that allows third-party 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. We will also create a use-case application that simulates a third-party developer that will use the payment framework to receive Bitcoin payments.

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 do not 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 [11], credit card details have to be loaded in only once. A parent can load a credit card for a child and let the child use it, but this can lead to problems of unregulated spending.

Table 1.1: Table of preferred online method of payment [6].

Credit Card	Debit Card	PayPal
48%	30%	12%

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

The last payment method we look at is a voucher based system, like Apple's Gift Cards. Vouchers are 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.

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 [10], but a small fee may be required for transactions that do not meet certain criteria discussed in chapter 2.

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 an international study done in 2014 [6], 48% of people prefer to make online payments with credit cards and 30% prefer debit cards, with 12% preferring PayPal with the remaining 10% using unspecified payment platforms. It is clear from these statistics that card payment methods largely dominate the online payments market.

Credit card payments are easy to make and are processed quickly. The costs of transactions are very difficult to determine, since they differ from bank to bank, the merchant and the user's specific credit card package. However, credit card payments can be inexpensive, 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. They can also provide insurance against fraud.

According to a survey done in 2014, a quarter of adults do not have bank accounts [12]. This makes it difficult for them to purchase anything online. Furthermore, there are prerequisites to getting a credit card, like minimum income. Credit cards also usually have a fixed monthly fee.

Table 1.2: Table of SMS payment payouts from bulksms.com [7].

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

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 traceable by the credit card company. This gives the credit card company the ability to track and profile users based on their spending [13].

Even though credit cards have small transaction fees, they still have a small fixed fee. This is an issue when payments are very small. For example, the company PayFast [14] charges a fixed R2 plus 3.9% of the payment as a fee. Therefore, when a user wants to make a R5 payment, the merchant will only receive R5 - R2 - R5 $\times$ 3.9% = R2.81.

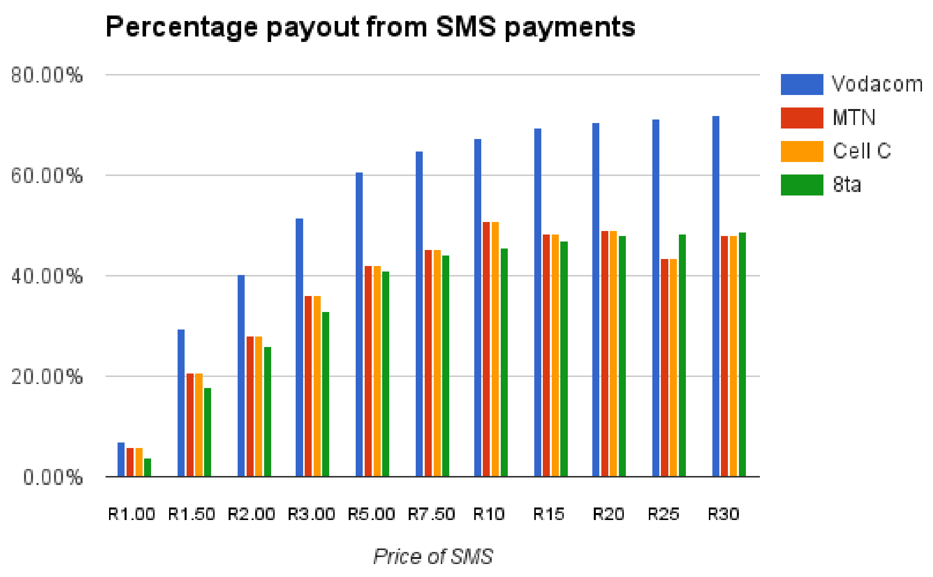
1.2.2 Airtime Payment

A popular mobile messaging application, Mxit, uses airtime payments with its Mxit Moola [15] 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%.

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.

Figure 1.1: Graph showing payout percentages that a merchant receives for SMS payments.



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 accessibility. 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.

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

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

Physical vouchers that are sold at stores are physical items that have to be manufactured and distributed across the world. This adds to the overhead cost, and it is also 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 chosen 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

To test the viability of Bitcoin, we will implement a Bitcoin payment system that runs on WeChat. This payment system should resemble a Bitcoin wallet application that allows users to send and receive Bitcoin.

The Bitcoin wallet's main objective will be to make payments within the WeChat application. Therefore, our main objective is to design the payment framework that other applications within WeChat will use to request and receive payments. This payment framework should also be able to confirm that a payment has been made.

We also want to test the practical limitations of making small Bitcoin transactions with low transaction fees. We want to investigate the effect of lowering the transaction fees and find a practical lower bound of transaction fees.

To implement our Bitcoin payments framework and wallet application, we will design a use-case application that also runs on WeChat. We will design and implement an application where gamebooks are sold and read. This application should present the user with a list of books that can be bought. The user should then be able to pay for the book by using our wallet application or any Bitcoin wallet application. If the payment is successful, the user should be able to read the gamebook on their phone.

Therefore, the objective is to be able to receive Bitcoin payments on a social media platform with smaller transaction fees than other payment systems.

Chapter 2

Literature Study

2.1 Bitcoin

Bitcoin is a protocol that was introduced in 2008 [2]. Bitcoin is a digital currency that allows a user to send money in the form of Bitcoin to any other Bitcoin address in the world. The transactions are verified by the Bitcoin network - a peer-to-peer network of computers or Bitcoin nodes that independently run Bitcoin software.

Bitcoin is decentralised which means that no country, organization or person controls it. It uses cryptography to remove trust from a central authority. With other payment mechanisms, a trusted third party is required to prevent double-spending [2]. Double-spending is a problem in digital payment systems that allows a malicious user to spend the same money twice. Bitcoin solves the double-spending problem by using a proof-of-work chain that can be trusted as long as the majority of the Bitcoin network is controlled by honest nodes.

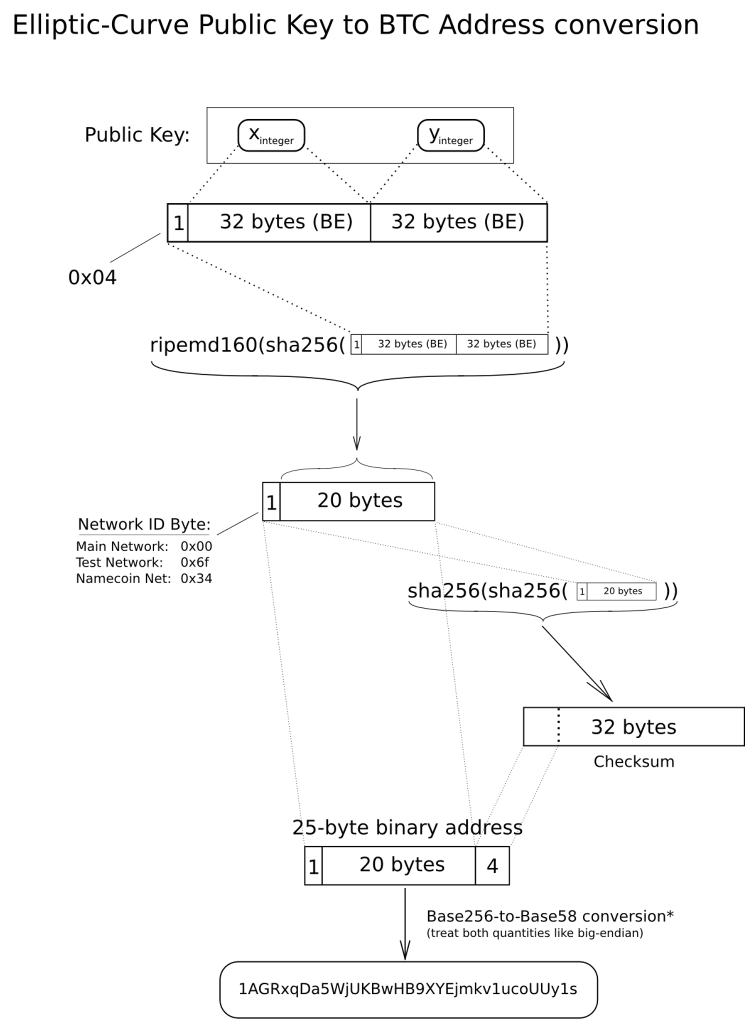
The code that is responsible for the Bitcoin network is open source and is maintained by a core team of developers. Since the code is open source, anyone can verify the code to ensure it is not malicious. Each node on the network chooses what code to run, but nodes must reach consensus by means of the majority of the network accepting or rejecting certain transactions.

2.1.1 A Bitcoin Address

A Bitcoin address is similar to a bank account that people can pay money to. An address is an alphanumeric string, like 1EZzmuCJqb6yhiowbdd7qgBQsiAJS4YEn3, that is used as a destination for a payment. An address is only an identifier of the wallet that it belongs to, and cannot be used to spend the Bitcoin that belongs to it.

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.

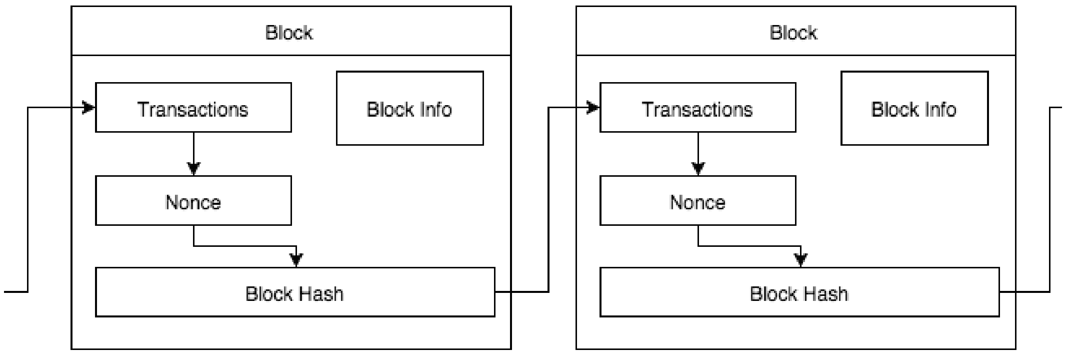
Figure 2.1: Public Key To Address Conversion. Adapted from [1].


Elliptic-Curve Public Key to BTC Address conversion
Public Key:
$x_{integer}$
$y_{integer}$
1
32 bytes (BE)
32 bytes (BE)
0x04
ripemd160(sha256(1 32 bytes (BE) 32 bytes (BE)))
1
20 bytes
Network ID Byte:
Main Network: 0x00
Test Network: 0x6f
Namecoin Net: 0x34
sha256(sha256(1 20 bytes))
32 bytes
Checksum
25-byte binary address
1
20 bytes
4
Base256-to-Base58 conversion*
(treat both quantities like big-endian)
1AGRxqDa5WjUKBwHB9XYEjmkv1ucoUUy1s

The public key is generated from the private key using the Elliptic Curve Digital Signature Algorithm [16] with parameters specified in the secp256k1 specification [17] [18] that was chosen for the Bitcoin protocol. The public address is generated from the public key with a series of hashes shown in figure 2.1. This allows the public key to only be made public once funds from an address are spent and not when receiving funds. This adds a layer of security to reduce risk.

An analogy of a postal box can be used to explain a Bitcoin address. The public address 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, they need the key. Therefore, the private key is like the key to the post box. Only the person with the key can open the box and retrieve the content. One important difference to a normal postal box is that the postal box is completely

Figure 2.2: Simplified representation of Bitcoin blockchain. Adapted from [2].



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

2.1.2 How Bitcoins Are Stored

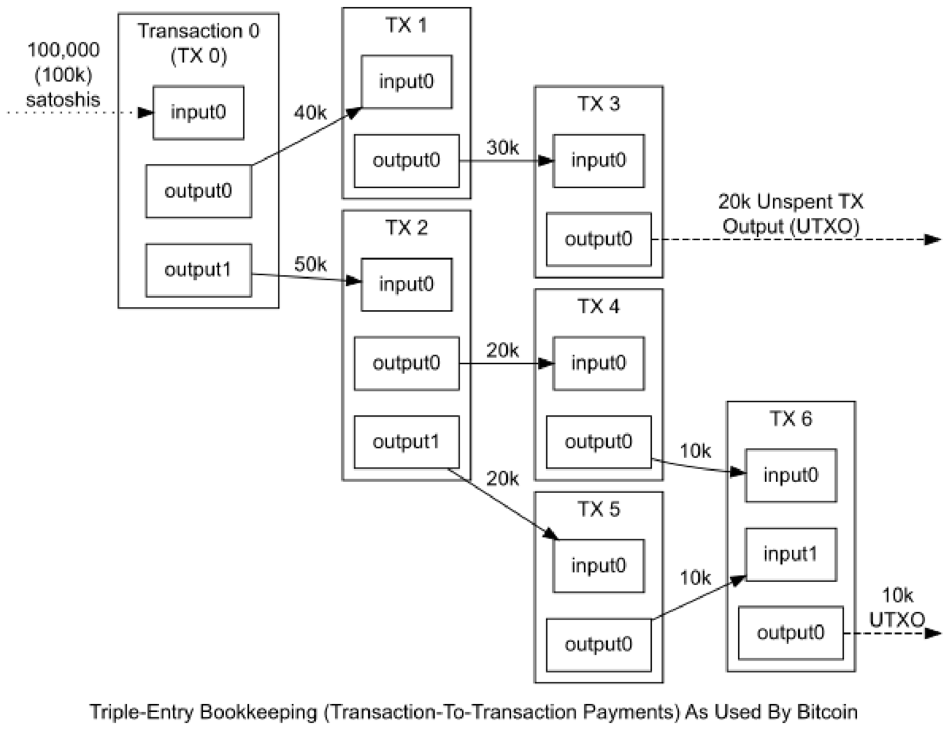
Bitcoin is "stored" using a public ledger called the blockchain. Effectively, every single successful Bitcoin transaction is stored in this public ledger, and every user that runs a full Bitcoin node has an identical copy of the blockchain. The authenticity of the blockchain is managed by using consensus 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. Every block's hash is the hash of the entire chain behind it as can be seen in figure 2.2. 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 is publicly available, it is possible to calculate the balance of any address at any given time. To store a Bitcoin wallet, only the private key of the wallet is required. A wallet is not a file containing Bitcoin. The ownership of the Bitcoin is recorded in the Blockchain. Even though the public address is derived from the private key, a server usually stores the public address as well, since it will be inefficient to calculate it everytime 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 two simple values securely, without having to keep track of complex databases 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. An input in a transaction is an output from a previous transaction to form a

Figure 2.3: Example of Bitcoin transactions demonstrating the relationship between inputs and outputs [3].



transaction chain. This output can then be used as a new input for a new transaction as can be seen in figure 2.3.

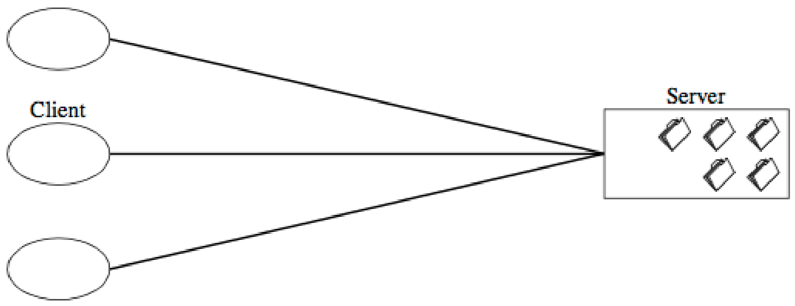
When looking at an address on the blockchain, we can determine which of the outputs of its transactions are not spent yet. They are called unspent outputs. These outputs can be cryptographically verified to belong to a specific address and can be used as inputs in a new transaction. The transaction then 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 by the payer, or the same address that the payment came from.

The difference between the sum of the inputs and outputs of a transaction is the transaction fee that is claimed by users that verify transactions. This is called mining and is discussed in section 2.1.4.

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 number of zeros. 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

Figure 2.4: Client-server Model of REST architecture [4].



using a brute force method. Therefore, more computation power increases the probability of finding a block that satisfies the zeros criteria.

The number of zeros required is called the “difficulty”, since more zeroes makes it more difficult to find a valid block. The difficulty 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 the most recent block, it has 1 confirmation. The longer the chain becomes after this block, the more confirmations it has and the transaction can with higher trust be regarded as final. Valid transactions that are published to the Bitcoin network are stored in each nodes memory pool (mempool) until they are mined into a block or discarded [3]. These transactions have 0 confirmations.

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 a double spend attack lowers exponentially [2]. With very small payments, it is not not required to have 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.

2.2 REST

Representational State Transfer (REST) is a Web architecture introduced in 2000 by Roy Fielding [4]. REST uses a client-server model to transfer data to and from the server. The interaction between the client and server is designed to be stateless. Therefore, every request to the server and every answer must contain all the information to complete the request. The stateless design induces visibility, reliability and scalability according to Fielding [4].

REST uses HTTP methods explicitly [8] to transfer resources to and from the server as described in table 2.1. Since standard HTTP methods are used, Web servers do not need to use specialised protocols to use a REST architecture. Furthermore, the REST service can be built using any backend webserver

Table 2.1: HTTP Methods used in the REST Architecture [8].

HTTP Method	**Usage In REST**
POST	Create resource on server
GET	Retrieve resource from server
PUT	Change or update resource
DELETE	Remove resource

software that is capable of receiving the HTTP methods from table 2.1, has a scripting language and preferably some sort of storage like a database.

REST uses URIs (Uniform Resource Identifiers) to distinguish method calls that are being made. The URI contains the names of specific resource that is required in a hierarchical structure. For example, the URI "http://www.myservice.org/discussion/topics/topic" would call the service at myservice.org with a specific topic in the topics category under the discussion category [8]. The curly brackets indicate a variable. Therefore, "topic" would be replaced with something like "sport". These URIs usually do not contain file extensions, like ".php", to be able to change the backend scripting language without changing the URIs.

2.3 Gamebooks

Gamebooks, Interactive Fiction and Choose Your Own Adventure are terms used for stories told in a non-linear way where the user is presented with choices that influence the flow and outcome of the story. The Choose Your Own Adventure brand of books was first published in 1979 after an author called Ed Packard aproached a small publisher in 1976 with a gamebook manuscript [5].

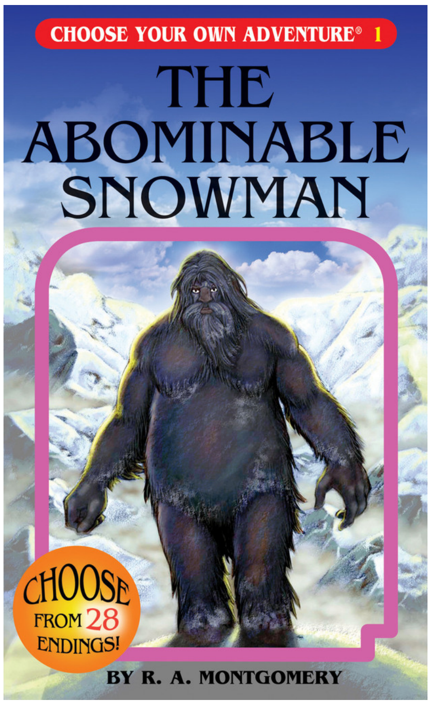
In a typical gamebook, the reader is given choices that are made by going to the page number indicated by the choice. For example, a reader can be given a choice of going through one of two doors. The choice can tell the reader to go to page 55 to go through the first door or to page 56 to go through the second door. If a gamebook is read only once, many pages of the book will not be read.

According to cyoa.com, gamebooks appeal to reluctant readers do to the interactive nature of gamebooks [5].

Gamebooks are also available electronically. A company called Choice of Games [19] is an example of an electronic gamebook publisher. Their gamebooks are available on a browser or on standalone mobile apps. They created a scripting language called "ChoiceScript" to allow authors to write gamebooks in an easy and intuitive way.

Since a gamebook is a piece of text with options as links, any website can easily be made into a gamebook. A website called Create Your Own

Figure 2.5: Example of one of the earliest Choose Your Own Adventure books [5].



Story [20] uses a MediaWiki [21] framework to write gamebooks that anyone can contribute to.

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 connected pieces.

A REST API will form the core of the framework. It will be responsible for managing payment request made by clients that use the framework.

The REST API will be accompanied by a wallet application that will be directly connected to the REST API. This will allow a user to reference a payment request from the wallet application to simplify the payment process for the user.

To practically test the payment framework, we will create a use-case application that will use the framework to sell virtual gamebooks.

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

3.1.1 REST API for payment management

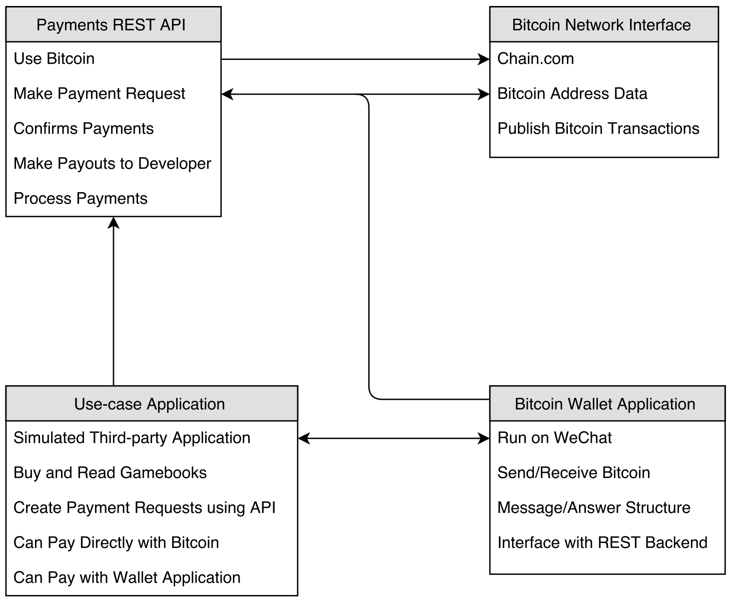
A REST (Representational State Transfer) API [22] was chosen for the main interface for developers to use Bitcoin without running a Bitcoin node or having experience with Bitcoin. REST was chosen because it is a commonly used architecture, it is easy to use and understand and it does not constrain the user's choice of programming language or environment.

An alternative to REST is SOAP (Simple Object Access Protocol) [23]. REST was chosen above SOAP due to the fact that REST uses existing generic web interfaces [4]. REST was also chosen due to the author's familiarity with the architecture.

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. Therefore, the system should generate a new Bitcoin address on request.

The payment requests that are made by developers are the resources of the REST API. A payment request will consist of a Bitcoin address to receive

Figure 3.1: Interaction between the parts of the system.


Payments REST API
Use Bitcoin
Make Payment Request
Confirms Payments
Make Payouts to Developer
Process Payments
Bitcoin Network Interface
Chain.com
Bitcoin Address Data
Publish Bitcoin Transactions
Use-case Application
Simulated Third-party Application
Buy and Read Gamebooks
Create Payment Requests using API
Can Pay Directly with Bitcoin
Can Pay with Wallet Application
Bitcoin Wallet Application
Run on WeChat
Send/Receive Bitcoin
Message/Answer Structure
Interface with REST Backend

payment and a payment identifier number to reference the payment. It will also contain information about the payment, like the amount to be paid, a lable that describes the payment and the status of the payment.

For simplicity sake, we will only use the HTTP methods POST and GET to create and retrieve resources, respectively. We will not implement the PUT and DELETE methods, since changing or deleting a payment resource is not necessary for the payment framework to function.

As mentioned in chapter 2.2, REST uses URIs to reference resources. A specific part of the URI that identifies the resource will be referred to as an "endpoint" in the rest of this chapter. The URI contains the domain name and other information that is used in every request that do not identify the specific request.

3.1.1.1 The /payment endpoint

The /payment endpoint 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 endpoint returns a Bitcoin public address and a payment ID.

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

3.1.1.2 /payment/{PUBLIC_ADDRESS} and /payment/{ID}

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

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

3.1.1.3 /balance

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

3.1.1.4 /payout

The /payout endpoint 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 into the platform, as well as an existing user base.

The Wallet Application will be directly connected to the back-end of the REST API. Every payment request is asociated with an unique ID. Therefore, payment requests will be accessible directly from the Wallet Application by using the ID in conjunction to referencing a Bitcoin address.

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 for this project, it will be quite expensive to run the Bitcoin software. According to a test done in July 2014 [24], running a full Bitcoin node Amazon

Web Services cost $42.06 in a month. The node used about 135GB of data transfer that cost $19.46. The data transfer is also dependant on the amount of Bitcoin transactions that are made, so the price may vary.

The price of running a Bitcoin node is acceptable when running a production service that requires it, but for the purposes of testing the viability of Bitcoin on a social media platform, we do not need a full Bitcoin node.

Therefore, 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 APIs.

Such a service must be able to get the balance and unspent outputs of a Bitcoin address. It should also be able to post a Bitcoin transaction to the Bitcoin network. For our testing purposes, this service must also be able to use the Bitcoin testnet. The Bitcoin testnet is an alternative blockchain that functions almost identically to the standard blockchain, but is designed not to have any value to allow developers to build Bitcoin systems without using real Bitcoin [25].

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 [5] 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 then query the API to confirm that the payment is received.

When the payment has been received, the user will then be able to read a gamebook that they bought using Bitcoin on their phone. The interaction between the Gamebooks application and the rest of the system can be seen in figure 3.1.

Accompanying the Gamebook application, we will design a Gamebook author webpage to allow people to write the gamebooks that will be available on the application. With this design, it would be possible for authors to write and publish gamebooks and be paid for their work by the readers that buy the gamebook.

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

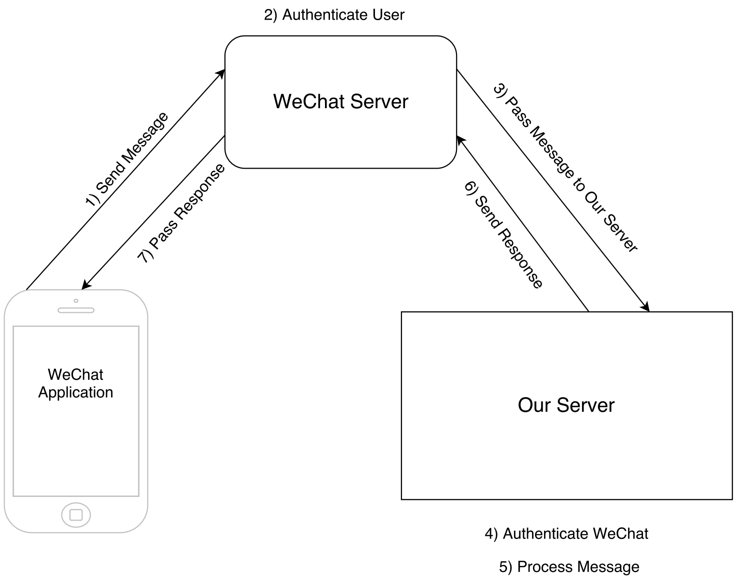
We decided to use XAMPP (Apache, MySQL, PHP and Perl) for the back-end development environment. XAMPP is a full stack development environment. It includes an HTTP server (Apache), a database server (MySQL) and a scripting language (PHP). XAMPP is free and easy to deploy, and is also used because the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



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.

Therefore, 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 specific Official Account. The identifier is not a global unique identifier for the entire WeChat platform. Therefore, 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. 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 [26]. 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.

Figure 4.2: Gamebook database table design

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

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 enables them to have circular stories and 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

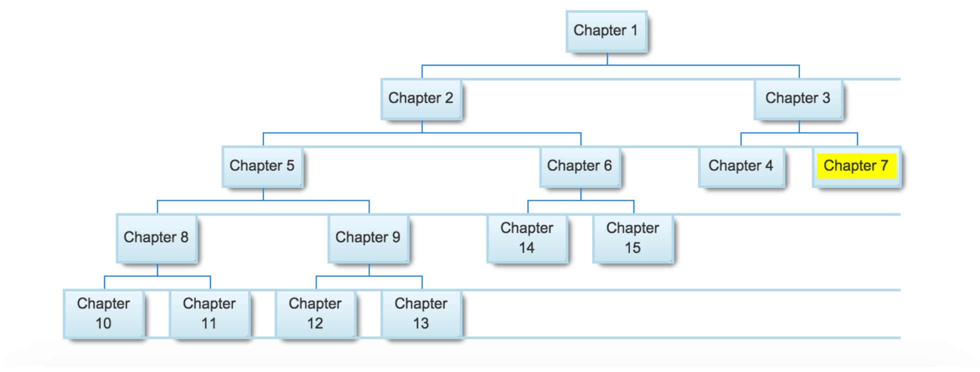
As stated in section 4.2.3, we are not going to use a scripting language to write the Gamebook in. It does not fit the design of the application that we are building, and has a learning curve for people that want to start writing books and are not familiar with scripting languages. We decided to use 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 with nodes representing a page.



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 does not 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 methods 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

We need to generate a Bitcoin address for each user. To do this, we use 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 that it does not 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 therefore 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 not 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.

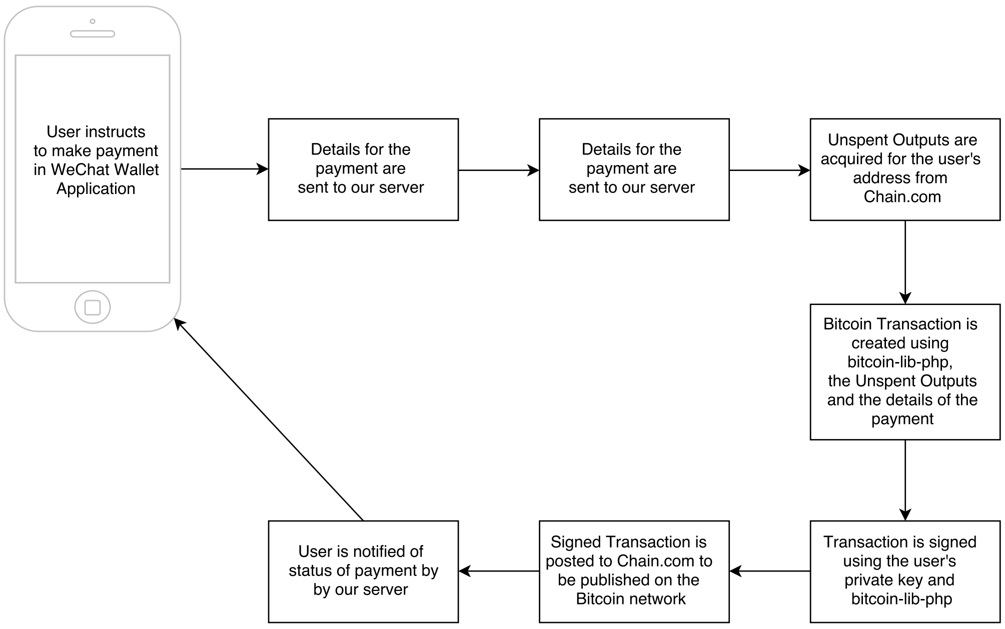
These unspent outputs are only acquirable with access to the Blockchain and we do not run the Bitcoin software. Therefore, we do not have direct access to these outputs. From our requirements in section 3.1.3, we require a third-party Bitcoin interface to provide these outputs among other things. From 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.

Figure 4.4: Bitcoin payment interaction between a user's phone and the payment framework.


User instructs to make payment in WeChat Wallet Application
Details for the payment are sent to our server
Details for the payment are sent to our server
Unspent Outputs are acquired for the user's address from Chain.com
Bitcoin Transaction is created using bitcoin-lib-php, the Unspent Outputs and the details of the payment
Transaction is signed using the user's private key and bitcoin-lib-php
Signed Transaction is posted to Chain.com to be published on the Bitcoin network
User is notified of status of payment by by our server

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 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. Therefore, 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 specification 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,

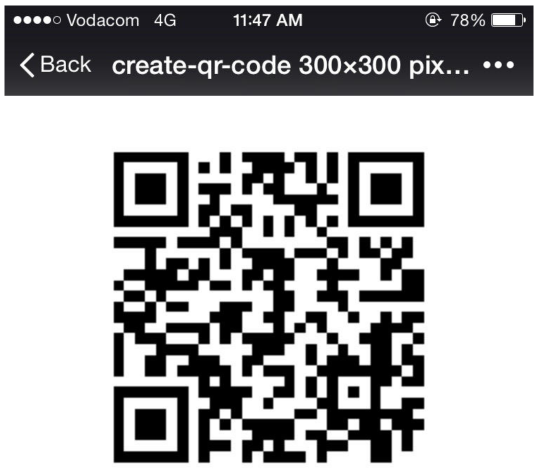


Figure 4.5: Example of a generated QR-code rendered in the WeChat browser.

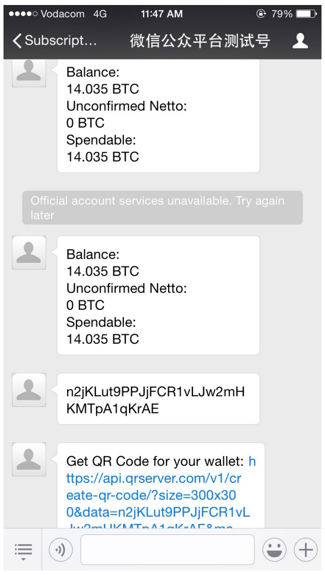
it does not allow the user to scan a QR-code directly from an Official Account. An Official Account does allow a user to send an image to our server. This will allow us to decode the QR-code on our server. For simplicity, we decided to use a third-party service called goQR.me to decode the QR-code. WeChat 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.

Figure 4.6: Example of the response for a balance request in the wallet app.



Balance:
14.035 BTC
Unconfirmed Netto:
0 BTC
Spendable:
14.035 BTC

Official account services unavailable. Try again later

Balance:
14.035 BTC
Unconfirmed Netto:
0 BTC
Spendable:
14.035 BTC

n2jKLut9PPJjFCR1vLJw2mH
KMTpA1qKrAE

Get QR Code for your wallet: h
ttps://api.qrserver.com/v1/cr
eate-qr-code/?size=300x30
0&data=n2jKLut9PPJjFCR1vL

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.

4.4 REST API Design

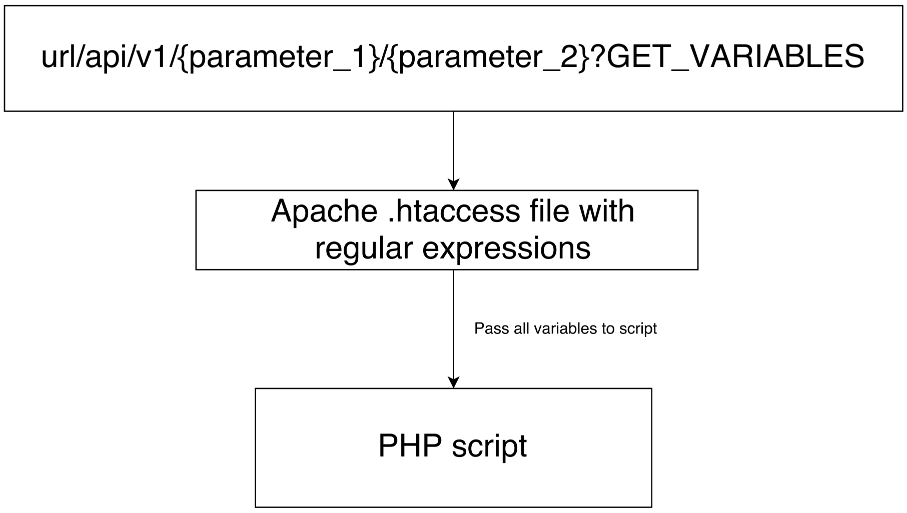
There are many ways to implement a REST API. This is because of the "black-box" approach of REST. As such, the first major design decision that needs to be made is whether to use a REST framework or to build it from scratch. Using a REST framework would be easier and standardized and if the framework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

Figure 4.7: REST URI routing.


url/api/v1/{parameter_1}/{parameter_2}?GET_VARIABLES
Apache .htaccess file with regular expressions
Pass all variables to script
PHP script

we are using an Apache server, we need to modify the ".htaccess" file with regular expressions.

Therefore, 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.

The PHP script will firstly authenticate the developer, followed by 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.

Table 4.1: Table of selected .htaccess flags.

Flag	**Description**
NC	No Case. Do not 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.

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.

Table 5.1: Table of selected HTTP status codes.

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

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 use 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

Table 5.2: Table of response times in seconds of our WeChat Official Accounts in comparison with an existing Official Account.

	Menu	**Browse**	**Balance**	**SuperSport**
	8.8	6.8	12.4	5.9
	7	6.4	8.5	6.3
	9.1	6.1	14.3	3.9
	6.1	6.6	7.1	5.8
	10.7	8.5	10.5	4.9
	4.7	8.3	7.6	6.1
	4.2	7.8	8	8.4
	6.4	7	7.3	5.2
	11.3	6.3	8.3	8.4
	6.6	6.1	8.1	7.9
Average	**7.49**	**6.99**	**9.21**	**6.29**
Std. Dev.	**2.40**	**0.90**	**2.41**	**1.52**

5.1.2 Timing Tests

To test the loading times, we sent messages to the Gamebooks and Wallet Application and measured how long it took to get a response. We conducted three loading tests:

1. Gamebooks Menu - minimal database queries
2. Browse Gamebooks - substantial database queries
3. Wallet Balance - substantial database and Bitcoin queries

In conjuction with testing our Official Accounts, we also tested a popular Official Account from the sport broadcast company SuperSport to have a baseline for comparison.

As can be seen from table 5.2, the loading times for our Official Accounts are very high. Getting the wallet balance took the longest of all the requests that we tested. The reason for the long time is all the steps that have to be completed for this request. First, the request is sent from the WeChat Application on the mobile phone. That request has to go to WeChat's server in China. If we ping wechat.com, we get an average round-trip time of 492ms. WeChat then has to validate the request and send it to our server in Singapore.

Our server then has to validate the request from WeChat and do database operations to get the address of the user. Since we are using the third-party service Chain.com for Bitcoin queries, we have to make a GET request to their services. This request has to check the Bitcoin Blockchain to determine the balance of the specified address. Checking the balance and the round trip of the request seems to drastically increase the time of the interaction, since the request that involved Bitcoin took roughly two seconds longer to complete. We used a webpage that we wrote to test the Chain.com API to test how long a "balance" call of an address takes, and found that it took an average of 2.5 seconds.

Fortunately, the timing tests of our baseline WeChat Official Account, SuperSport, produced very similar results to our tests. SuperSport performed 10% faster than our average fastest test and 32% faster than our average slowest test. Although the SuperSport Official Account performed faster than ours, it still performed very slow compared to a dedicated webpage loading time. The average loading time that we measured on webpage running on the same server as the backend of the Wallet and Gamebooks Apps was 2.7 seconds. This loading time is drastically shorter than the loading time of the requests in WeChat, and as such we can conclude that WeChat's Official Accounts are not the most ideal platform for speed. A web-app will outperform WeChat based on the tests that we have done.

5.1.3 Transaction Fees and Time Testing

To test the viability of Bitcoin with regards to transaction fees and the time the transaction takes to be confirmed, we will measure how long transactions take to complete. We will start with the default transaction fee (0.0001 BTC) and halve the fee every transaction. The time a transaction takes to complete is nondeterministic and is dependant on many factors discussed in chapter (insert reference here).

An initial test was done by measuring the time taken since a transaction was posted on the Bitcoin network to when it was included in a block (one confirmation). This turned out to be a badly constructed test, since it measured the time it takes in minutes rather than the number of blocks that was mined since the transaction was posted. This is an important distinction, because the time a block takes to mine is not dependant on the details of the transactions that are being mined.

We record the current block height at the time the transaction is posted and then compare it to the block height of the block that the transaction is included in. In the ideal situation, a transaction will be included in the next block that is mined and will have a block difference of one.

For all of the Bitcoin fees tests we use the value of R3685 for 1 Bitcoin taken from a South African Bitcoin exchange, BitX, on 19 October 2015. Every payment we make is roughly R1 worth in Bitcoin. The defualt fee of 0.0001 Bitcoin is roughly 37 cents. With a R1 transaction, the fee default fee is 37% of the value. This fee is also added on top of the payment, and is not covered by the merchant like with airtime or credit cards. Therefore, if a user makes a R1 payment with Bitcoin to a merchant, the user will effectively pay R1,37.

Table 5.3: Table showing the results of a test to show the relationship between the transaction fee and the time it takes to complete the transaction.

Amount (BTC)	Fee (BTC)	Fee (ZAR)	Block Difference	Time
0.00027137	0.0001	0.3685	1	0:20:13
0.00027137	0.00005	0.18425	1	0:01:55
0.00027137	0.000025	0.092125	1	0:00:55
0.00027137	0.0000125	0.0460625	8	1:48:14
0.00027137	0.00000625	0.02303125	85	15:06:00
0.00027137	0.00000313	0.01153405	16	2:00:56
0.00027137	0.0000028	0.010318	6	0:39:57

For this reason we want to test by how much we can feasibly reduce the fee that the user pays.

Since making the transaction fee smaller will reduce the average time transactions take to be confirmed, we have to define an acceptable transaction time. For practical purposes, we define this time as 24 hours. Therefore we want a transaction to be confirmed within 24 hours, but due to the small transaction we can provide the virtual goods or services as soon as a valid transaction is posted to the Bitcoin network.

In table 5.3 we can see that the longest transaction took 85 blocks and 15 hours to be confirmed. This is an interesting transaction, since the next two transactions were completed considerably faster, even though they had smaller fees. This highlights the difficulty of accurately predicting how long a transaction will take to be confirmed. The number of factors determining the transaction time are too high, and the random element of mining increases it further.

Therefore, this test does not provide a metric, but rather shows that it is possible to make Bitcoin payments with very small fees that are confirmed within reasonable time.

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 User 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.

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.

Table 5.4: Table of ratings of individual parts of the system.

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

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 User 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 third-party application. Most of the users were very optimistic about the concept of using Bitcoin as payment on the on a social platform. The majority also like the Gamebooks application in conjunction with the Author page.

The most negative feedback came from the response that only 40% of users said that they would prefer to use the Wallet Application over traditional payment mechanisms, even though 60% said it is easier and 50% said it is faster. From user's responses, it seems that many of the users do not trust Bitcoin as a payment mechanism. Many users also disliked the user interface of the WeChat platform and also complained that responses are very slow.

Chapter 6

Conclusion

In the area of small mobile payments on a social media platform, Bitcoin was found to be a viable payment mechanism. Bitcoin was shown to be able to process a transaction with a fee of about R0.01 on a R1 transaction, where the same transaction with an SMS payment would cost at least R0.90.

With the payment framework that was implemented on WeChat, we were successfully able to manage Bitcoin payments as a service. The Bitcoin walllet application that was built to run on WeChat was able to function as a basic Bitcoin wallet application. It was also able to make payments directly to the use-case gamebooks application.

Despite the constrained environment of the chat application, the user feedback for the overall experience of using Bitcoin as a payment mechanism was positive. Further constraints, like relying on third-party services that increase response time increased the negative feedback regarding the user experience.

6.1 Contributions

6.2 Future Work

The implementation of the payment framework in this project is intended to be a proof of concept. As such, there was not enough focus on security, especially in terms of the storage of Bitcoin private keys. If a framework like this is implemented on a production platform, extreme attention should be given to security and possible exploitations.

Furthermore, a platform like the one built would be better if it relied less on multiple third-party platforms for services like Bitcoin blockchain access and QR-code decoding. For the sake of security and control, services like this would be better to be implemented by the organisation that builds the platform. Having control of a bigger part of the system reduces the points of failure.

Appendices

Appendix A

No appendix yet

Bibliography

[1] "File:PubKeyToAddr.png." [Online]. Available: https://en.bitcoin.it/wiki/File:PubKeyToAddr.png

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

[3] Bitcoin.org, "Bitcoin Developer Guide." [Online]. Available: https://bitcoin.org/en/developer-guide

[4] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Building*, vol. 54, p. 162, 2000. [Online]. Available: http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm

[5] Cyoa.com, "History of CYOA."

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

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

[8] IBM, "RESTful Web services: The basics." [Online]. Available: http://www.ibm.com/developerworks/library/ws-restful/

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

[10] Bitcoinwiki, "Transaction Fees." [Online]. Available: https://en.bitcoin.it/wiki/Transaction_fees

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

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

[13] Visa, "Privacy policy." [Online]. Available: https://usa.visa.com/legal/privacy-policy.html

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

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

[16] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[17] B. Wiki, "Secp256k1." [Online]. Available: https://en.bitcoin.it/wiki/Secp256k1

[18] C. Research, "Standards for Efficient Cryptography 2 (SEC 2) : Recommended Elliptic Curve Domain Parameters," vol. 2, no. Sec 2, pp. 1–33, 2010.

[19] C. of Games, "Choice of Games." [Online]. Available: https://www.choiceofgames.com/

[20] Create Your Own Story, "Create Your Own Story." [Online]. Available: http://editthis.info/create_your_own_story/Main_Page

[21] Mediawiki.org, "Welcome to MediaWiki.org."

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

[23] D. Box, D. Ehnebuske, A. Layman, N. Mendelsohn, L. D. Corp, and H. F. Nielsen, "Simple Object Access Protocol (SOAP) 1 . 1 Table of Contents," no. May, pp. 1–26, 2000.

[24] P. Halliday, "Running a full Bitcoin node on AWS." [Online]. Available: http://pghalliday.com/aws/bitcoin/2014/05/02/running-a-full-bitcoin-node-on-aws.html

[25] Bitcoinwiki, "Testnet." [Online]. Available: https://en.bitcoin.it/wiki/Testnet

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