

# **Pay with Nano: A Cryptocurrency Payment Processor**

*Zack Zixu Xiang*

Master of Science  
Computer Science  
School of Informatics  
University of Edinburgh  
2018



# Abstract

Cryptocurrencies lack widespread adoption while the existing payment processing systems charge high fees. This project aims to address both problems by designing and implementing *Pay with Nano*, a cryptocurrency payment processor that utilises the instant, fee-less, and scalable Nano cryptocurrency. The payment channel itself inherits the fee-less nature of Nano to allow the merchant to accept electronic payments without processing charges. The payment processor is accompanied with a set of easy-to-use user interfaces designed for the general public. *Pay with Nano* also supports sales statistics, refunds, and simultaneous payments. Additionally, it does not store the user's password or the wallet seed, thus giving the user the full responsibility of safekeeping the information. This sense of autonomy aligns with the ownership idea that is embedded in cryptocurrencies. *Pay with Nano* comes as a responsive web application that functions on devices of different screen sizes. It is powered by a Flask server developed using the Python programming language.

The performance of the system is assessed by performing test payments that simulate real-world settings. It has achieved a median payment confirmation time of 9.6 seconds and a median settlement time of 28.2 seconds. Both times are expected to be reduced by deploying the application onto a cloud server. The usability of the application is evaluated by conducting a Think Aloud study that exposes a number of issues and its corresponding potential improvements.

# Acknowledgements

First of all, this work would not be possible without my supervisor, Dr Christophe Dubach. I sincerely thank him for his patient guidance and insightful comments. Every meeting was fruitful and had left me deep in thoughts.

I wish to acknowledge the help provided by the Nano discord community. They have patiently answered my questions and pointed me to valuable resources.

I would like to express my deep gratitude to all of my friends who had supported me throughout this journey. In particular, I would like to thank Sotiris Nanopoulos for the numerous constructive conversations that had taken place in the library. His expertise on Blockchain and cryptography are greatly appreciated. I would also like to offer my special thanks to Michiko Cruz for generously spending her time to proofread this dissertation. I wish to also thank Mohamed Amine Belabbes. His one late-night check-up call has earned him a spot in the acknowledgement. That is time well spent. There are many more friends who I cannot individually name. Thank you all. I am sure you know who you are.

Last but certainly not least, I would like to thank my family for all the support provided throughout the years, and for shaping me into who I am today.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

*(Zack Zixu Xiang)*

Dedicated to my beloved family

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	A Brief History of Money . . . . .	1
1.2	The Money Problem . . . . .	2
1.2.1	Value . . . . .	2
1.2.2	Ownership . . . . .	3
1.3	The Engineered Money . . . . .	3
1.4	Payment Processor . . . . .	5
1.5	Problem Definition . . . . .	6
1.6	Contributions . . . . .	6
1.7	Report Structure . . . . .	7
<b>2</b>	<b>Background</b>	<b>9</b>
2.1	Nano . . . . .	9
2.1.1	Block Lattice . . . . .	9
2.1.2	Representatives . . . . .	12
2.1.3	Fork Resolution . . . . .	12
2.2	Wallet . . . . .	13
2.2.1	Deterministic Account Generation . . . . .	13
2.2.2	Wallet Management Service . . . . .	14
2.3	Flask Micro-framework . . . . .	15
2.4	Think Aloud . . . . .	16
<b>3</b>	<b>Related Work</b>	<b>17</b>
3.1	Fiat Payment Processor . . . . .	17
3.2	Cryptocurrency Payment Processor . . . . .	18
3.3	Comparisons . . . . .	19

<b>4</b>	<b>Overview</b>	<b>21</b>
4.1	Specifications . . . . .	21
4.2	Assumptions . . . . .	22
<b>5</b>	<b>Front-end Design</b>	<b>25</b>
5.1	Familiarity . . . . .	25
5.2	Simplicity . . . . .	26
5.3	Security . . . . .	28
5.4	Responsiveness . . . . .	30
<b>6</b>	<b>Back-end Design</b>	<b>31</b>
6.1	Information Delivery . . . . .	31
6.2	Payment Flow . . . . .	32
6.3	Payment Processor Design . . . . .	33
6.4	Refund . . . . .	35
6.5	Security . . . . .	36
6.6	Account Recovery . . . . .	36
6.7	System Recovery . . . . .	37
<b>7</b>	<b>Implementation Details</b>	<b>39</b>
7.1	Full Stack Overview . . . . .	39
7.1.1	Database . . . . .	39
7.1.2	Nano Node . . . . .	40
7.1.3	Back-end . . . . .	40
7.1.4	Front-end . . . . .	42
7.2	Development Methodology . . . . .	42
<b>8</b>	<b>Evaluation</b>	<b>45</b>
8.1	Performance Evaluation . . . . .	45
8.1.1	Payment Time . . . . .	45
8.1.2	Recovery Testing . . . . .	48
8.2	Usability Study . . . . .	48
8.2.1	Participant Selection . . . . .	48
8.2.2	Methodology . . . . .	49
8.2.3	Results . . . . .	49
8.2.4	Solutions . . . . .	50



<b>9 Conclusion</b>	<b>51</b>
9.1 Critical Analysis . . . . .	52
9.2 Future Work . . . . .	52
<b>A Think Aloud Researcher Script</b>	<b>55</b>
<b>B Think Aloud Tasks</b>	<b>57</b>
<b>C Think Aloud Consent Form</b>	<b>59</b>
<b>D Think Aloud Questionnaire</b>	<b>61</b>
<b>Bibliography</b>	<b>63</b>



# List of Figures

1.1	The number of non-cash transactions (in billions) from 2000 to 2016. Figure taken from [1]. . . . .	5
2.1	Visualisation of the Nano block lattice. <i>A</i> , <i>B</i> and <i>C</i> represent three different accounts and their corresponding account-chains are shown. Every transfer consists of a send block (S) and a receive block (R). The time when blocks are added to the block lattice is taken from a non-interacting observer node. Figure taken from [nano-white]. . . . .	11
2.2	A graphical representation of the hierarchical structure of a Nano wallet. Texts in grey present an arbitrary example. . . . .	14
2.3	The encoding lookup table for Nano addresses. Figure taken from the Nano discord group. . . . .	14
3.1	The steps required to confirm a payment is successfully made when using a fiat payment processor. . . . .	17
4.1	The flowchart of all required steps for one payment cycle. The double-lined rectangle indicates a front-end change. The normal rectangle represents a back-end process. The rounded rectangle represents a user action. The dashed border indicates that the application is not responsible for this step. . . . .	24
5.1	The login (left) and the registration (right) pages. . . . .	25
5.2	The page presented to the new user when he first logs in. . . . .	26
5.3	The POS terminal. . . . .	27
5.4	The payment page presented to the customer. . . . .	28
5.5	The presentation of the the possible final outcomes of the payment, where (a) indicates payment successful and (b) indicates unsuccessful. . . . .	28

5.6	The home screen of the dashboard. . . . .	29
5.7	The graphical fingerprint representation is displayed beside the welcome text. . . . .	30
5.8	The payment page presented on a smartphone (left), a tablet (middle), and a laptop (right). . . . .	30
6.1	An overview of how the server obtains the required information. In this diagram, each rectangle represents a separate component. A rounded rectangle represents a persistent storage entity. Rectangles with dashed border are not under the project's responsibility. . . . .	31
6.2	An example of the payment flow. $C_i$ represents the account of the $i^{th}$ customer. $M_j$ represents the receiving account of the $j^{th}$ merchant. $T_l^k$ represents the $l^{th}$ transition address associated with $k^{th}$ merchant. Funds flow in the direction of the arrow. . . . .	33
6.3	The architecture of the payment processing components. The arrows represent the direction of the information flow. A rounded rectangle represents a persistent storage entity. The peripheral components, such as the refund service and the authorisation service, are not included in the diagram. . . . .	34
8.1	Histogram of the completion time of the payment phase, gathered from 100 runs. . . . .	46
8.2	Histogram of the completion time of the settlement phase, gathered from 100 runs. . . . .	47
8.3	Histogram of the total time spent for one transaction cycle, gathered from 100 runs. . . . .	47

# Chapter 1

## Introduction

Money is the bedrock of the human civilisation. It defines a measure of value and facilitates the exchange of goods and services among parties. Yet, despite its tremendous importance, rarely does one ask - "What makes money... money?" This question was first formally explored by William Stanley Jevons, who was also accredited for introducing mathematical methods into economics. In his book *Money and the Mechanism of Exchange* [2], Jevons outlined seven properties of the ideal form of money: utility, portability, indestructibility, homogeneity, divisibility, stability of value, and cognizability.

This introductory chapter seeks to motivate the need for a cryptocurrency payment processor by taking a deep dive into the topics of money, cryptocurrencies, and payment processing. The seven properties are used as the framework for comparison throughout. The chapter then defines the problem, outlines the contributions, and finally concludes with a structure of the report.

### 1.1 A Brief History of Money

For millennia, gold was the most widely-accepted form of money. It was the material that most aligned with the properties above. Gold was hard to find. Even at present days, it requires a significant and ever-increasing amount of resources to mine, thus limiting the rate of new supply introduced into the market. Its scarcity retains its value, through the vast span of human history.

However, gold is not the ideal form of money. Most obviously, it lacks portability. The inconvenience becomes evident when the transaction amount is sufficiently large.

The second generation of money comes in the form of fiat currency, which is currently what most of us referred to as "money". Its value is derived from the promise of the issuing government, hence the name "fiat", translated to "it shall be" in Latin [3]. The currency itself could be cheaply produced by the government and easily transferred, resulting in a drastic improvement on portability and divisibility in comparison to gold. Furthermore, the coming of the information age brings along the digitalisation of fiat currency and other classes of assets, thus eliminates most of fiat currency's shortcomings, such as its lack of indestructibility.

It seems that the digitalised fiat currency is the perfect medium to facilitate the value transfer between individuals and to fuel the country's economy.

Until it isn't.

## 1.2 The Money Problem

### 1.2.1 Value

Value, as previously discussed, is arguably the most important property of money. However, for fiat currency, its possessed value is controlled by a central authority. The scarcity of money is only superficial, as the government has the ability to freely increase the money in circulation. In a relatively stable country, well-calculated monetary injections lead to a controlled rate of inflation, which in turn stimulates consumer spending and economic growth. Consequently, a side effect is the continuous devaluing of its fiat currency over time. The Federal Reserve of the United States, for example, aims to induce 2% inflation per year [4], while the purchasing power of the US dollar has dropped by half in under 30 years [5]. The stability of the value of the US dollar is only short-term.

The ability to produce more money could lead a country into dire situations if the government does not have its citizen's interests at heart. The fact is, almost all governments in the world are borrowers. Therefore, devaluing its own currency is also effectively loosening its debt burden. On the other hand, those who hold the currency, i.e. the citizens, lose value in their savings. Essentially, a high rate of inflation results in a forced wealth transfer from the citizens to their government.

History has proven many times that many governments cannot resist the temptation of inducing such forced wealth transfer and the rate of inflation could quickly spiral out of control. The hyperinflation of Zimbabwe, where the peak

inflation rate was at 98% per day [6], rendered many people's savings worthless. At the time of writing, an economic crisis is ongoing in Venezuela, where the inflation rate is on course to reach 1 million per cent by the end of 2018 [7]. When people no longer trust that the country's fiat currency still has value tomorrow, the currency no longer has the intended utility. That money is no longer good money.

### 1.2.2 Ownership

The digital transformation adds convenience to our lives but in turn sacrifices ownership. When we deposit money into a bank, for example, we receive only a proof of deposit in return. The bank then takes control of the money and updates the account balance in the database which it also controls. This account balance, however, does not guarantee the availability of the money. The bank could refuse to facilitate a transaction due to a number of reasons, such as the amount involved is too large. We no longer have the full ownership of our own money.

The same could be said for other physical assets, such as gold. Presently, most of the gold is locked inside various vaults. Buying gold usually only represents purchasing the ownership agreement of a piece of gold - a piece of gold you would never touch.

## 1.3 The Engineered Money

It is then not difficult to extrapolate that in order to fulfil the seven properties of money while not sacrificing ownership, the type of asset has to be purely digital. Blockchain, the end product from the dovetail of cryptography, distributed computing, and mechanism design, has made it a reality. The concept of blockchain was first published by S. Nakamoto [8], along with the first practical implementation of what is now called cryptocurrency – Bitcoin. Anyone can send any value of Bitcoin to anywhere in the world through a decentralised peer-to-peer (P2P) network. Anyone can also truly own Bitcoin by simply owning the private key to the corresponding wallet.

For the first time in the history of mankind, we have engineered money.

The underlying value of such an asset was quickly realised. In merely 7 years, the price of a Bitcoin has increased by more than 300,000 times, from \$0.06 in

2011 to \$19,000 at the December 2017 peak [9]. However, it was also during this peak where the fundamental flaws of Bitcoin were exemplified. The technology was not ready for this level of mass adoption.

The first issue is scalability. The current Bitcoin network could include roughly 2000 transactions per block [10]. During the peak, a premium transaction fee is required to ensure that your transaction is processed. The average transaction fee skyrocketed, reaching as high as \$28 per transaction [11]. As a block is processed every 10 minutes, we could derive that the current Bitcoin network could handle up to 7 transactions per second at maximum capacity - far inferior to other payment networks, such as Visa, which could process up to 24,000 transactions per second [12].

The second issue is latency. Disregarding the 1,188 minutes average waiting time during the peak demand period, it still normally requires an average of 20 minutes to confirm a transaction. This is far longer than the acceptable waiting time for most of the everyday transactions, such as buying coffee. As a result, merchants, who accept Bitcoin, are forced to accept zero-confirmation transactions and are exposed to the risk of the replace-by-fee double spend attack if not careful [13].

The final and arguably the most concerning issue is the network's power consumption. At the end of November 2017, the network was consuming more electricity than 159 countries [14], and the estimated power consumption has already doubled 5 months later [15]. This exponential growth in power consumption will soon induce a significant impact on the environment. The growth is also expected because of Bitcoin's *Proof-of-Work* (PoW) consensus system, where the participants are economically incentivised to use more energy to compute more hashes, thus receiving a larger share of the reward.

These flaws have led to the development of numerous alternative cryptocurrencies. Nano, formerly known as RaiBlocks, is one of the contenders to Bitcoin. It has promised instantaneous, fee-less and power-efficient transactions, with theoretically unlimited scalability. If the claim is true, then it has achieved the ultimate portability, both in term of time and cost.

There is obviously one property that Bitcoin, Nano and other cryptocurrencies are lacking - stability in value. With its predictable emission rate and transparent total supply, any cryptocurrency has the potential to reduce its volatility to that of gold. What is missing is general acceptability. When the general public agrees



that a cryptocurrency has value and uses it in everyday lives, then its price is no longer derived from the speculation of its utility but the actual utility it possesses. The stability in value naturally follows.

## 1.4 Payment Processor

Payment processing service has become a necessity in the modern world. The European Central Bank estimates that in the Eurozone alone, the total number of non-cash payment per year has reached 122 billion in 2016 [1]. Figure 1.1 shows the increasing popularity of the electronic non-cash payment methods.

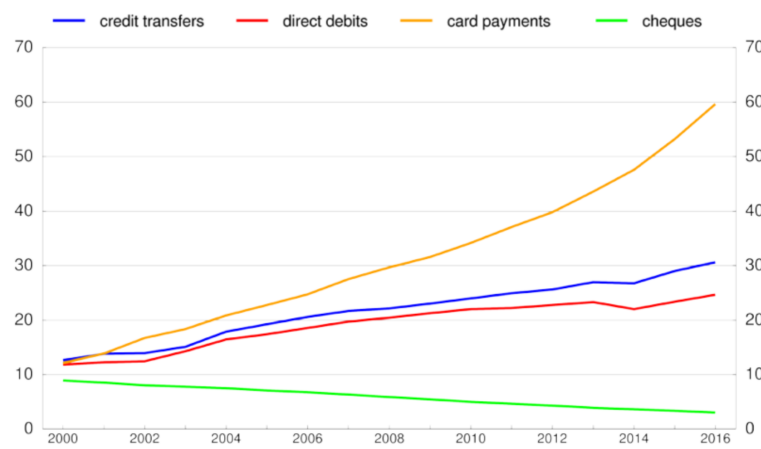


Figure 1.1: The number of non-cash transactions (in billions) from 2000 to 2016. Figure taken from [1].

However, the spread of electronic payment systems did not stop the payment processing companies and networks from charging high fees. In the UK, the most popular method for accepting non-cash payment is through WorldPay. WorldPay currently charges a fee of up to 2.75% of the transaction value, a subscription fee of £19.50/month, and a terminal rental fee of £17.50/month [16]. This barrier prevents many small independent businesses from accepting electronic payments, which results in losing potential sales.

Furthermore, the public places their unquestioned trust in these middle-men. Their security measures and reliable services are almost taken for granted, and severe damage would occur when they fail to meet these expectations. In June 2018, the Visa network experienced a European-wide outage, resulting in 5.2 million failed transactions and countless lost man-hours [17]. A later analysis

suggested that this crisis was caused by a switch failure. It is then reasonable to speculate that there may be other single points of failure that exist in the payment processing pipeline, and any of them could result in a complete halt of the pipeline.

## 1.5 Problem Definition

The issues stated above engender an alternative means of accepting non-cash payments for products and services, which utilises the power of cryptocurrencies, the superior form of money. They call for a payment channel that is fee-less, fast, and secure. Nano is then the perfect medium of exchanged for this channel. The channel should require the minimum level of trust from both the customer (payer) and the merchant (payee). Additionally, it should not have control over any user's fund, and this aligns with the ownership idea central in cryptocurrencies. This is what the project aims to provide - an alternative that offers significant advantages over the status quo.

A broader problem with the cryptocurrencies is the lack of adoption in everyday scenarios. To solve this, we need easy to use applications that target the general public, and a good cryptocurrency payment processor is certainly one of them. Thus, this project, featuring a user-friendly interface that a non-tech-savvy merchant could easily pick up, also aims to help address the general acceptability problem of cryptocurrencies. It aims to contribute to shaping an alternative global economy powered by a currency that is open, neutral, and transparent – a currency that is not controlled by anyone but owned collectively by everyone.

## 1.6 Contributions

This project has produced the following contributions:

- *Pay with Nano*, a fully-functional cryptocurrency payment processor, powered by the Nano network, is designed and implemented. It features a Point-of-Sale (POS) terminal and a dashboard. *Pay with Nano* is the pioneer solution that allows merchants to accept Nano at the checkout of brick-and-mortar stores or service professionals to accept Nano from clients on-site,

and it is the only Nano payment processor that supports peripheral services, such as refunds.

- A novel approach of monitoring the Nano network is purposed. It addresses the unreliability issue of the current callback implementation while keeping the resource usage low.
- The performance and the usability of the payment processor is evaluated, after completing the implementation. An evaluation report is produced and included in this thesis.

## 1.7 Report Structure

The structure of the dissertation is as follows:

Chapter 2 discusses the theoretical aspects of Nano, including the block lattice structure and fork resolution. It also examines the Nano wallet management service in detail. It then discusses the Flask micro-framework and concludes by describing the Think Aloud protocol.

Chapter 3 features the payment processors, currently existing in the market, that cater to both fiat money and cryptocurrencies. It concludes by showing the advantages of *Pay with Nano* over these competitors.

Chapter 4 presents the specifications in detail, where the exact steps to complete a payment cycle is shown. It also lists out the assumptions that the project is based on.

Chapter 5 explains the front-end design choices made for the application. It discusses how the user interface obtains certain desired characteristics.

Chapter 6 exposes the back-end design, which includes how the payment processor is designed at the module level, how its security is ensured, and how the system recovers from an unexpected failure.

Chapter 7 lists the concrete technical details of the implementation, such as the specific tools and libraries used.

Chapter 8 reports on the evaluation results. It exhibits a performance evaluation and a usability study report from a Think Aloud session.

Chapter 9 summarises what is achieved in this project. It presents a critical analysis that reflects on both the good and bad decisions made when working on this project. It concludes by discussing the future works that may be explored.



# Chapter 2

## Background

### 2.1 Nano

Nano, former known as RaiBlocks, is proclaimed to be "an instant, zero-fee, scalable currency" [18]. It is driven by the philosophy of "do one thing and do it well", aiming to be the best cryptocurrency for everyday use.

#### 2.1.1 Block Lattice

The Nano whitepaper [19] proposes an augmentation to the Bitcoin's blockchain [8] structure by extending into two dimensions and creating the so-called "block lattice" structure. In essence, each account in the Nano network has its own blockchain, henceforth known as "account-chain", where only the account owner can add a new block.

Similar to a block on the Bitcoin blockchain, every block on an account-chain requires the previous block's hash, a digital signature, and a valid PoW. PoW is a moderately hard cryptographic puzzle where the account owner needs to find a valid solution. The solution is a nonce (unsigned integer) such that the hash of the previous block's hash concatenated with the nonce is less than a target. The target is inversely proportional to the puzzle difficulty [8]. In the case of Nano, the PoW puzzle is relatively easy, as its only purpose is to prevent transaction spamming. This is analogous to the *hashcash* email system [20], in which PoW is used to prevent email spamming.

There are four types of blocks available, each corresponding to an essential action defined in the protocol and carries a different set of information [21]:

- **Account creation.** It requires an *open* block, which should be always be the first block of any account-chain and have a previous block hash of zeroes. The block instead requires a source transaction hash, which means that an account can only be opened after receiving the fund from another account.
- **Sending.** This action requires a *send* block. The block specifies the account to which the fund is sent.
- **Receiving.** A *receive* is required to claim a sent fund. Each *receive* block must link to a *send* block by specifying the latter's hash. Note that the receiving action could be done at any time, thus making asynchronous account-chain updates possible. Additionally, if multiple *send* blocks are pending, the account owner could choose the order to receive the fund.
- **Assigning a representative.** This is done via a *change* block. A representative's duty is to ensure network consensus by voting on the behalves of the accounts under his representation. This is further explored in Sections 2.1.2 and 2.1.3.

Each block also records the chosen representative and the resulting account balance after the action specified by the block is performed. Thus, the entire account information is encoded into each block. This design allows quick account balance computation and aggressive pruning [22]. The design is only recently implemented into the protocol [23].

The blocks on an account-chain are linked via hash pointers, and each transaction could be thought as a *send* block extending an edge to the corresponding *receive* block, connecting two account-chains and establishing a block lattice structure. An example is illustrated in Figure 2.1. The causality of sending and receiving forbids the latter from happening before the former, thus the block lattice could be thought of as a directed acyclic graph (DAG) [24] with respect to the time axis. Navigating backwards in time from any point on the block lattice arrives at the genesis block, which is the special block that has started the whole block lattice. Note that Nano's DPoS consensus mechanism is radically different from Iota's Tangle [25], albeit both are categorised as DAG-based projects.

In Bitcoin's blockchain, a new block is produced roughly every 10 minutes. One transaction is considered to precede another if the former resides in an earlier block. Nano, on the other hand, sacrifices this sense of time. The asynchronous

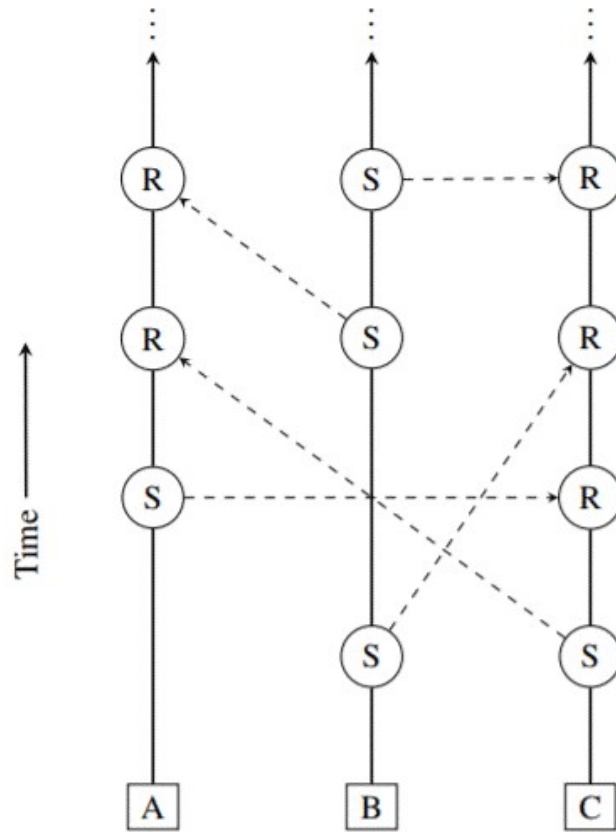


Figure 2.1: Visualisation of the Nano block lattice.  $A$ ,  $B$  and  $C$  represent three different accounts and their corresponding account-chains are shown. Every transfer consists of a send block (S) and a receive block (R). The time when blocks are added to the block lattice is taken from a non-interacting observer node. Figure taken from [nano-white].

nature of the network means that there is no universally agreeable way to know which transaction happened first if two transactions are made from two different account-chains. Considering the two transactions sent to account  $A$  in Figure 2.1 for example, in the frame of reference of the observer node,  $C$  has broadcasted its *send* block before  $B$ . However, there is also a possibility that  $B$  has broadcasted before  $C$ , but is heard later due to the network latency.

Sacrificing the notion of time and transaction ordering allows Nano to achieve near instantaneous transaction speed. It does, however, forbid Nano from introducing any time-based features, such as timelocks [26], at the protocol level.

### 2.1.2 Representatives

The Nano network's consensus protocol is based on Delegated Proof-of-Stake (DPoS) [27], where representatives are elected to maintain network consensus. The exact method of establishing consensus is examined in Section 2.1.3. An account becomes a representative when other accounts, with a cumulative Nano balance of 256 or more, assign it as the representative. The weight of a representative's vote is equal to the total amount of Nano under its representation.

Representatives are expected to run a full node that is online all the time, but unlike the traditional DPoS model, they are not rewarded for maintaining consensus, allowing Nano transactions to be fee-less. At the first glance, it seems that there are no incentives for anyone to become a representative, but there are external factors to be considered. First of all, running a Nano node is famously cheap. It could cost as little as \$3/month to run a node on a cloud server [20]. It is then not hard to imagine local businesses running nodes, and the money saved from not paying fees to the traditional payment processors would quickly offset the small cost. Second, many online services could set up representatives and nodes as a cheap alternative form of advertisement and trust-building. These services include Nano ledger explorers, web wallets, and exchanges. Whether or not the incentives are sufficient is beyond the scope of this project. Nevertheless, at the time of writing, there are 554 representative accounts, located all over the world, that are eligible to vote. The account with the most voting power is run by the Binance exchange, controlling 23% of the total voting weight [28]. Nano is, by every definition, a fully decentralised cryptocurrency.

### 2.1.3 Fork Resolution

In the customer-merchant context, the most relevant form of attack is the double spending attack where the malicious customer would attempt to spend the same fund more than once. In the Nano network, this action would result in a fork on the account-chain, where two or more *send* blocks claim to succeed the previous block. The network then relies on the representatives to provide a resolution.

When a representative node detects this abnormality, it calls for a 16-second "announcement round". During the announcement round, each representative places his vote on one of the blocks and broadcasts this information [29]. An observing node listens to the broadcasts and tallies up the votes. If the voting



weight for one of the candidate blocks reaches above 50% of the total online voting weight, then a quorum is achieved. The node could then safely write the winning block onto the ledger and discard the rest. If a quorum is not achieved, the announcement round will rerun three more times, and the block with the most voting weight at the end is chosen as the winner.

Unlike other block types, each new *send* block is automatically subject to an announcement round. This level of scrutiny means that once the *send* block passes the announcement round, the node could be confident that the block is valid and is already acknowledged by the network. In the context of this project, the merchant could confirm that the customer has genuinely paid even if he is yet to claim the payment using a *receive* block, thus reducing the waiting time.

## 2.2 Wallet

### 2.2.1 Deterministic Account Generation

This section outlines steps required to generate a set of deterministic accounts from a wallet seed [30]. Figure 2.2 presents a graphical representation of the steps.

The wallet seed is a random 32-byte value which only the wallet owner should know.

From the seed, a series of secret keys are produced. Each secret key is determined by hashing the seed concatenated with the position index. The hashing algorithm used is Blake2b [31].

A public key could then be produced from a secret key by feeding the secret key through the Ed25519 key generation function [32]. To finally arrive at a Nano address, the public key is divided into 5-bit chunks and encoded according to the scheme shown in Figure 2.3. Note that ambiguous character pairs, such as "0" and "o", are not included in the encoding alphabet. The encoded value is prefixed with "xrb\_" and suffixed with a checksum to give the final Nano address.

In general, a typical Nano user would be the owner of one wallet. From this wallet, he could then generate many Nano addresses and become the owner of multiple accounts.

The term "account" and "address" are used interchangeably in this report.

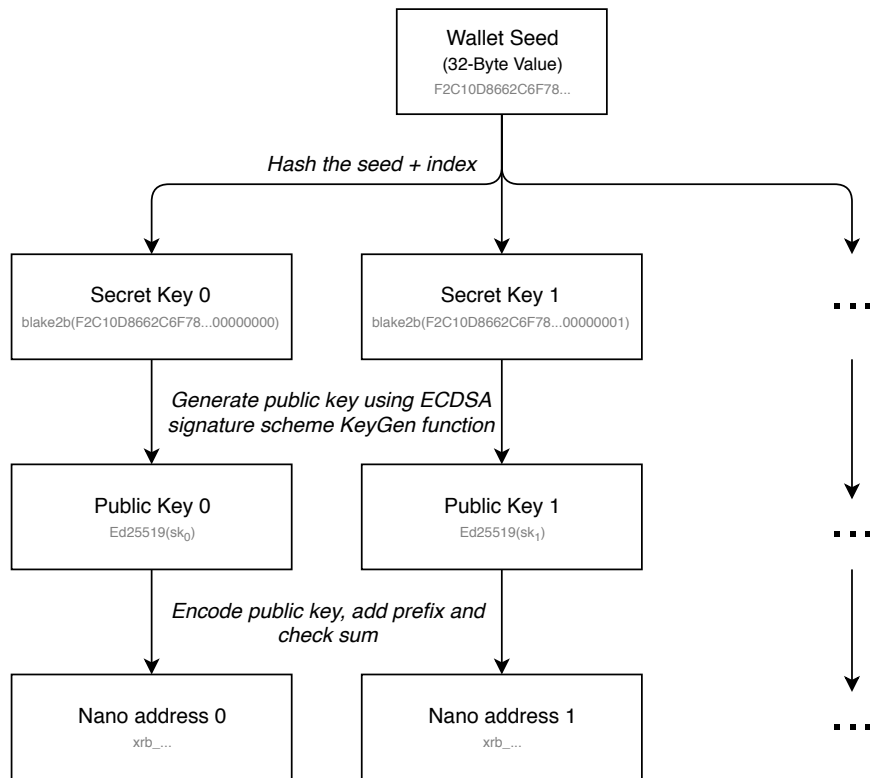


Figure 2.2: A graphical representation of the hierarchical structure of a Nano wallet. Texts in grey present an arbitrary example.

Binary	Encoded	Binary	Encoded	Binary	Encoded	Binary	Encoded
00000	'1'	01000	'a'	10000	'i'	11000	'r'
00001	'3'	01001	'b'	10001	'j'	11001	's'
00010	'4'	01010	'c'	10010	'k'	11010	't'
00011	'5'	01011	'd'	10011	'm'	11011	'u'
00100	'6'	01100	'e'	10100	'n'	11100	'w'
00101	'7'	01101	'f'	10101	'o'	11101	'x'
00110	'a'	01110	'g'	10110	'p'	11110	'y'
00111	'b'	01111	'h'	10111	'q'	11111	'z'

Figure 2.3: The encoding lookup table for Nano addresses. Figure taken from the Nano discord group.

### 2.2.2 Wallet Management Service

The official distribution of the Nano node comes with a wallet management service that fully complies with the design above. It also supports password encryption of the wallets.

A valid wallet file contains a randomly generated cryptographic salt. The service then encrypts the wallet seed with the password and the salt, using Advanced Encryption Standard (AES) [33], and the wallet is "locked". To check if a given password is valid, the service may try to "unlock" the wallet by decrypting

the encrypted seed using the password, but the service implements a better solution. During the initial locking process, it first encrypts the zero-value with the given password and the salt and stores the result as the verification value. When verifying a password, it again encrypts the zero-value with the password and the salt and checks if the result matches with the verification value. This password verification method is safer as the plain seed is not recovered during the process.

Finally, it is worth noting that the wallet management service generates a wallet identifier for each wallet. This identifier is used to index to the corresponding wallet information and is only stored locally.

## 2.3 Flask Micro-framework

Web application frameworks provide tools and libraries to simplify server-side web development. They abstract away lower-level networking primitives to allow the developer to work directly with HTTP requests and responses. They generally also support simplified routing and Object Relational Mapping [34] (ORM) that provide useful abstractions for accessing relational databases.

Flask is a Python web application micro-framework [35]. The term "micro" highlights its design philosophy, which is to be lightweight and customisable. Its vanilla version contains only the essentials for running a web server. Many libraries are built for Flask to provide additional features and can be integrated easily due to their "plug-and-play" design. For example, the *Flask-SQLAlchemy* extension provides Flask with the ORM feature.

Another Python web application framework that is often compared with Flask is Django [36], and they have polar-opposite design philosophies. Django is a "battery-included" framework that comes with tools to address many of the common problems out of the box. It usually has a "correct way" to solve a problem.

In contrast, Flask encourages a developer to consider what problems are to be solved and pick tools specific for these problems. The developer also has to make more design decisions when using Flask. Thus, it can be suggested that Flask offers more fruitful learning opportunities.

## 2.4 Think Aloud

The Think Aloud protocol is a qualitative method for evaluating the usability of an application [37]. It involves the participant verbalising their thoughts and actions while interacting with the interface. The task for the researcher is then to carefully listen. The benefit of a Think Aloud study is that the researcher can see how the participant misinterprets design elements, and more importantly, the research can hear *why* the participant has misinterpreted them. These are hugely beneficial for the next design iteration. Additionally, as each participant's interactions and thoughts are captured and studied in detail, it is thought that only five participants are needed to expose all of the major problems [38], making Think Aloud a cheap and quick evaluation method.

A problem with Think Aloud is that it does not provide any form of quantitative data. One might naively think that metrics, such as time-on-task and number of clicks are useful to record, but these are heavily inflated due to the fact that the participant is constantly talking, and they do not reflect the normal usage scenarios.

# Chapter 3

## Related Work

This chapter discusses the existing payment processor solutions for both fiat and cryptocurrencies. They are then compared with *Pay with Nano*.

### 3.1 Fiat Payment Processor

All fiat payment processors follow the same steps to accept and settle a payment. A full payment cycle consists of a payment phase, where a payment is accepted, and a settlement phase, where the payment arrives onto the merchant's account.

The payment phase is designed to be executed quickly, as both the customer and the merchant need to wait for it to complete. Figure 3.1 illustrates the steps required in this phase.

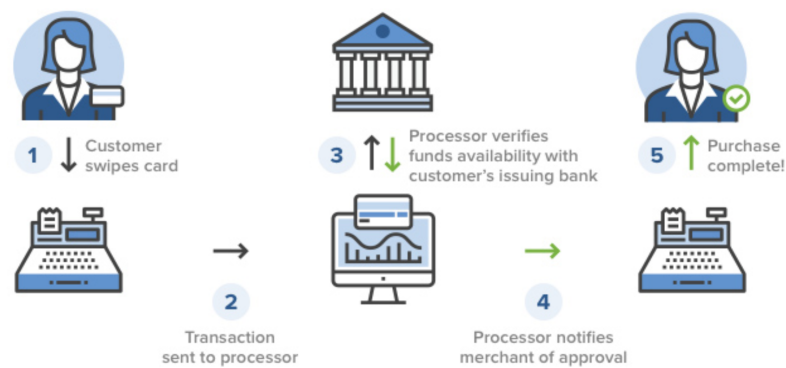


Figure 3.1: The steps required to confirm a payment is successfully made when using a fiat payment processor.

As soon as the customer presents his card details, in this case by swiping, the

details of the transaction are immediately forwarded to the payment processor. The payment processor then contacts the card issuing bank to check that the customer indeed has enough fund or credit available. Once the fund availability is confirmed, the payment processor displays a "success" message to the customer and the merchant, thus concluding the payment phase.

The settlement phase begins with the card issuing bank reimbursing the payment processor for the purchased amount. At the end of each day, the merchant sends a list of sales to the payment processor. The payment processor then deposits the sum into the merchant's bank account, and this step may require an additional day.

As previously mentioned, the most popular payment processor is WorldPay. It charges a fee of 2.75% for a credit card payment and 0.75% for a debit card payment, plus additional monthly charges.

As the FinTech industry develops, a number of challenger payment processing solutions have emerged. One that is often in the spotlight is Square. It offers simple and stylish card readers and Point-of-Sale (POS) solutions with no strings attached, accompanied by easy installation and intuitive user experience. It charges a fee of 1.75% of the transactional value per sale [39].

## 3.2 Cryptocurrency Payment Processor

A number of solutions have been developed to allow merchants to accept cryptocurrencies. The most popular of which is BitPay [40]. BitPay supports Bitcoin and Bitcoin Cash payments and converts them into fiat currency at settlement. It thus provides the full package that enables the merchant to accept cryptocurrencies just as another payment method, without needing to understand the concept of cryptocurrencies at all. Of course, if the merchant does have a Bitcoin wallet, he may choose to settle a percentage of the revenue in Bitcoin. BitPay charges a 1% fee and settles the merchant's earnings the next day.

Coinbase Commerce [41] is another cryptocurrency payment processor. In contrast to the full package offered by BitPay, Coinbase Commerce is effectively a wrapper of a cryptocurrency wallet that accepts Bitcoin, Bitcoin Cash, Ethereum, and Litecoin. It provides a payment interface that directs the customer to pay to an address generated from the wallet and presents receipts after the payment has been completed. The merchant is responsible for keeping the seed safe. This

aligns with the ownership idea of cryptocurrencies. However, the problem here is that each payment is stored on a different address, and withdrawing all of the payments would require a transaction from each address, which could amount to significant fees. Additionally, if paid in Bitcoin or Bitcoin Cash, the fund needs at least six confirmations before it could be spent, resulting in a settlement time of one hour.

### 3.3 Comparisons

The solutions mentioned above share one clear flaw - long settlement time. Additionally, the fees could be too high for the added convenience. In Table 3.1, these payment processing solutions are compared with *Pay with Nano*. Illustrating the monthly fee comparison is a hypothetical business that accepts £10,000 worth of digital transaction every month, consisting of 500 £20 payments. Amongst the payments, two-thirds are made with a debit card while a third is done with a credit card if a fiat payment processor is used. This ratio is chosen to reflect the UK debit and credit card payment volume figures provided in [42].

*Pay with Nano* is ultimately a service that enables free and immediate value transfer, in a world where information transfer has long been free and immediate.

Table 3.1: A comparison of payment processors.

Payment processor	Accepted currencies	POS terminal types	Settlement time	Cost for £10,000 monthly sales
WorldPay	fiat currencies	card reader	up to 48 hours	£179.10
Square	fiat currencies	card reader	up to 48 hours	£175.00
BitPay	Bitcoin, Bitcoin Cash	smartphone apps, web, card reader integration	up to 24 hours	£100.00
Coinbase Commerce	Bitcoin, Bitcoin Cash, Ethereum, Litecoin	web	up to 1 hour	various network transaction fees
<i>Pay with Nano</i>	<i>Nano</i>	<i>web</i>	<i>&lt;1 minute</i>	<i>Free</i>



# Chapter 4

## Overview

### 4.1 Specifications

The main objective of this project is to implement a payment processing application for the merchant to receive fee-less Nano payments from customers. The merchant initiates a payment request on the POS interface. The core interaction is described in Figure 4.1. Note that as far as the customer and the merchant are concerned, the payment is completed once the *payment phase* ends, and the application is ready for the next customer. The final payment claim requires the merchant's wallet application to broadcast a *receive* block; hence, it is not within the responsibility scope of this application.

Additionally, the payment processor should support the following features.

- **Different currencies:** Allow the merchant to input the amount in a currency of their choice. Convert it dynamically into the equivalent amount in Nano, using a live-price API.
- **Dashboard:** A dashboard that displays statistics and the transaction history, which can also be downloaded. It should also support the changes in account settings, such as the merchant's receiving address.
- **Refund:** Allow the merchant to refund any previous successful payment.
- **Responsiveness:** Allow the user to accept payments on different devices with various screen sizes.
- **Simultaneous payments:** Allow the user to initiate multiple payment requests at the same time, on the same device or on different devices.

Security is always the top priority during implementation. The system should be safe from both external attackers, who would try to compromise the system, and malicious customers, who would try to perform ill-intended actions when not under the merchant's supervision. Once security is ensured, ease-of-use becomes the focus. The design should minimise the number of steps required to perform an action and the number of choices at each step. At any time, it should be intuitive what should be done for both the merchant and the customer.

Finally, a design philosophy is imposed on the payment processor - the system should require the minimum level of trust from the user (the merchant). The system should not have any means to access the user's refund wallet. Ergo, the design embraces the concept of full-ownership that is embedded in cryptocurrencies, as only the user holds the private key to the refund wallet. In the case where the system has to temporarily withhold funds, the fund should be forwarded as soon as possible. Minimising the required trust also has the added benefit of minimising the responsibility of the system. This is further explained in Section 6.5.

## 4.2 Assumptions

An important assumption for this project is that the customer either pays nothing, which results in a timeout, or the exact amount requested. This premise is thought to mimic the behaviour of an electronic card payment, where only the exact amount must be paid. It is worth noting that a Uniform Resource Identifier (URI) standard is described in the Nano protocol and is supported by all Nano wallets. Therefore, the customer can scan the QR code to pay the exact amount using any wallet of choice.

The security of the system is based on the following premises:

- The Nano protocol is theoretically secure, and the corresponding implementation is sound.
- The wallet management service provided by the node is secure.
- The Python programming language and Flask framework are secure.
- External API calls are secure, and the values returned are trusted, under the assumption that the communication is done over HTTPS.

- The host machine, the browser, and the user's computer are secure unless already compromised.

These assumptions are, of course, not necessarily valid at all times. However, confirming the validity and attempting to resolve any security problem related to these assumptions are out of the scope of this project.

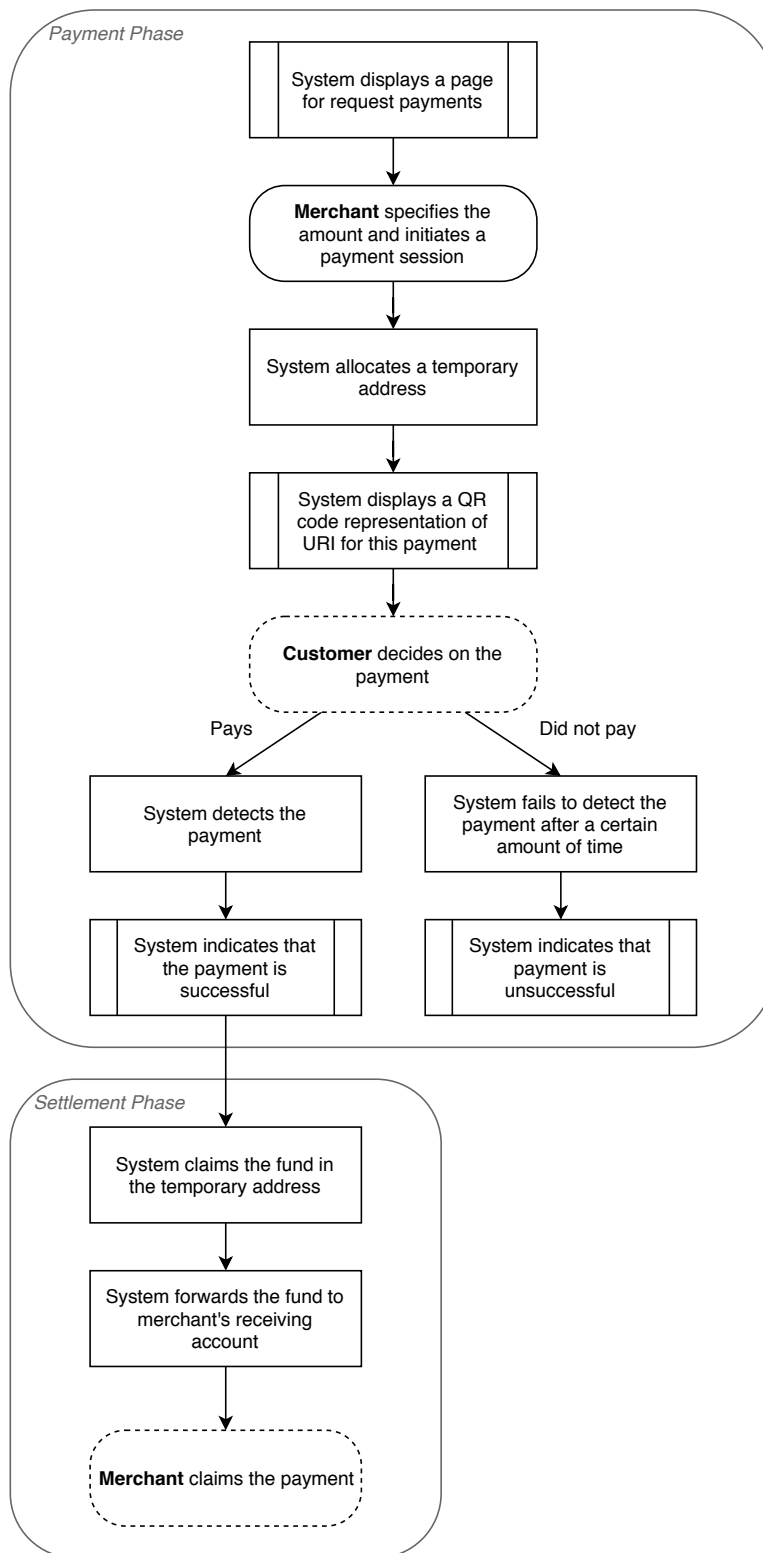


Figure 4.1: The flowchart of all required steps for one payment cycle. The double-lined rectangle indicates a front-end change. The normal rectangle represents a back-end process. The rounded rectangle represents a user action. The dashed border indicates that the application is not responsible for this step.

# Chapter 5

## Front-end Design

This chapter explains the rationale behind many of the design choices made when creating the front-end interface of the application. It also contains several examples of the visual elements of the application to further the explanation.

### 5.1 Familiarity

Humans are instinctively great at pattern matching. Given an interface that is similar to what they have previously experienced, they would quickly discover how to navigate a new one. Therefore, one of the design ideas is to abstract the interface away from the likely foreign concept of cryptocurrency and provide elements that introduce familiarity. This is exemplified in the authentication pages as shown in Figure 5.1

The figure displays two side-by-side web forms for authentication. The left form, titled 'Login Form', contains input fields for 'Username' and 'Password', a 'Log in' button, and a 'Recover from seed' link. Below these is a link for 'New to site? Create Account' and a 'Pay with Nano' logo. The right form, titled 'Create Account', contains input fields for 'Username', 'Email', 'Password', and 'Repeat Password', a 'Sign up' button, and an 'Already a member? Log in' link. It also features a 'Pay with Nano' logo at the bottom. Both forms have a clean, minimalist design with light gray borders and a white background.

Figure 5.1: The login (left) and the registration (right) pages.

The registration page, following the standard design pattern [43], requires the user to enter a username, an email, and a password. Similarly, the login page requests for a username and the corresponding password. Although in the back-end, the way that these credentials are used for registration and authentication completely deviates from the common practice, the difference is not revealed to the user.

Only when the user logs in for the first time would he discover that there are additional unfamiliar steps required to complete the setup. This design intentionally separates the familiar from the unfamiliar and gives the user an easy task to start and a slightly challenging one to discover afterwards. The user should never be too far away from his comfort zone.

## 5.2 Simplicity

Scott Belsky, the Chief Product Officer of Adobe, famously said: "Rule of thumb for UX: More options, more problems."<sup>1</sup> This design takes this quote to heart by eliminating all the unnecessary options.

Continuing from the previous section, if the user fails to complete the profile setup, he will be directed to the settings page, shown in Figure 5.2, once logged in.

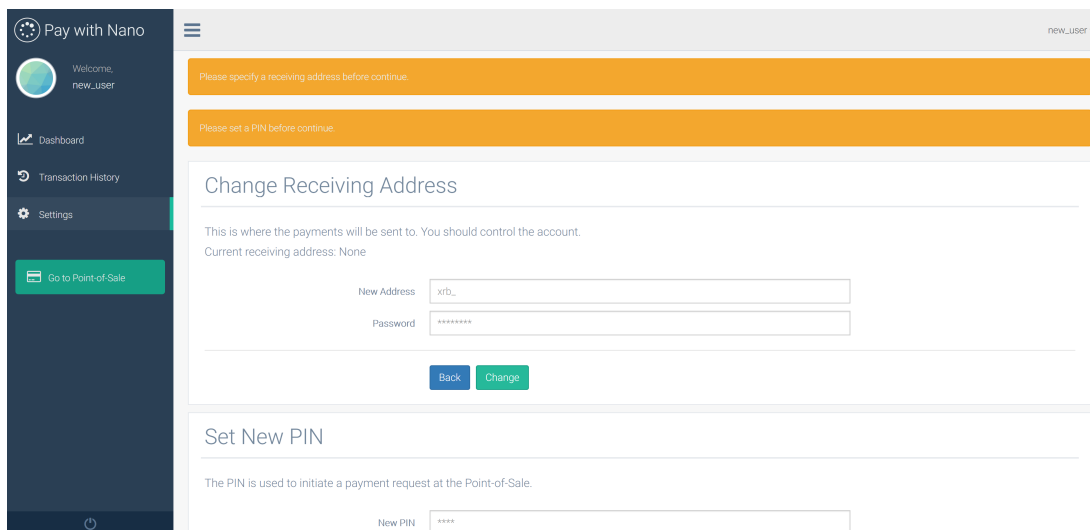


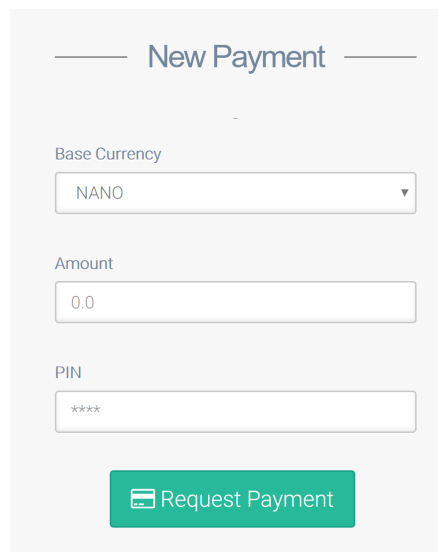
Figure 5.2: The page presented to the new user when he first logs in.

The user is prompted with alerts on what needs to be done, and the user has no option to leave the page until the task is complete.

<sup>1</sup><https://twitter.com/scottbelsky/status/504017374681378816>

The interface for payments is designed to be minimalistic and binary. When the user starts the POS terminal, it is launched in the new tab, which provokes the idea that the terminal is a separate entity from the dashboard. It also gives the user the ability to launch multiple POS terminals on the same device.

The home page of the POS terminal contains only one form, as shown in Figure 5.3, which asks for the details of the payment request. The user has two choices: to fill out the form and initiate the payment request or to leave the POS terminal by closing the tab. Once the payment request is initiated, the customer is presented with Figure 5.4 and also has two choices: to pay or not to pay, which leads to Figure 5.5(a) or Figure 5.5(b), respectively. Proceeding from this page redirects the user back to Figure 5.3. Thus, a payment request is completed.



The image shows a web form titled "New Payment". It contains three input fields: "Base Currency" (a dropdown menu currently showing "NANO"), "Amount" (a text input field showing "0.0"), and "PIN" (a masked text input field showing "\*\*\*\*"). Below these fields is a green button with a card icon and the text "Request Payment".

Figure 5.3: The POS terminal.

Another aspect of simplicity is to minimise the number of steps required to perform a task, without clogging up the page. This idea is already demonstrated in the payment request process described above, but a more prominent example is how the user could launch the POS terminal from the dashboard. This is thought to be by far the most frequent operation to be performed on the dashboard and therefore should be the most reachable.

As shown in Figure 5.6, a bright coloured button for launching the POS terminal is fixed on the left sidebar, thus allowing the user to perform this action with one click anywhere within the dashboard. It could also be seen that hyperlinks to changing the receiving address and making refunds are placed in the appropriate sections on the home page. While these actions are important,

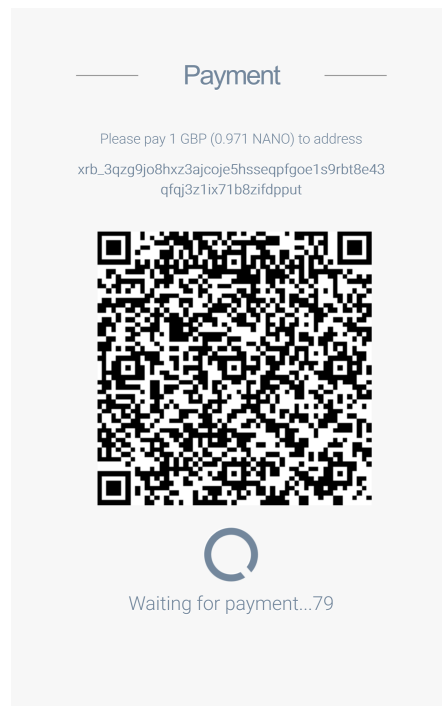


Figure 5.4: The payment page presented to the customer.

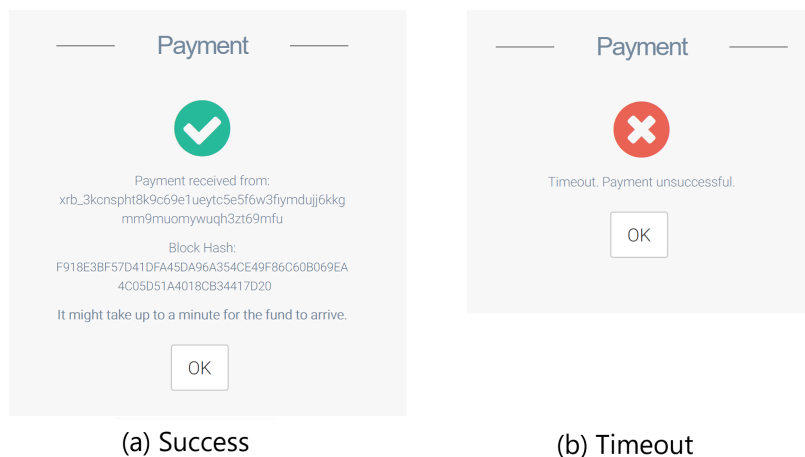


Figure 5.5: The presentation of the the possible final outcomes of the payment, where (a) indicates payment successful and (b) indicates unsuccessful.

these requests are thought to be performed less frequently. They are set aside as they require more steps, ensuring that the home page is kept clean and tidy.

## 5.3 Security

The interface employs the idea of defensive design [44], where it is assumed that a malicious customer would try all possible ways to misuse the application. The goal



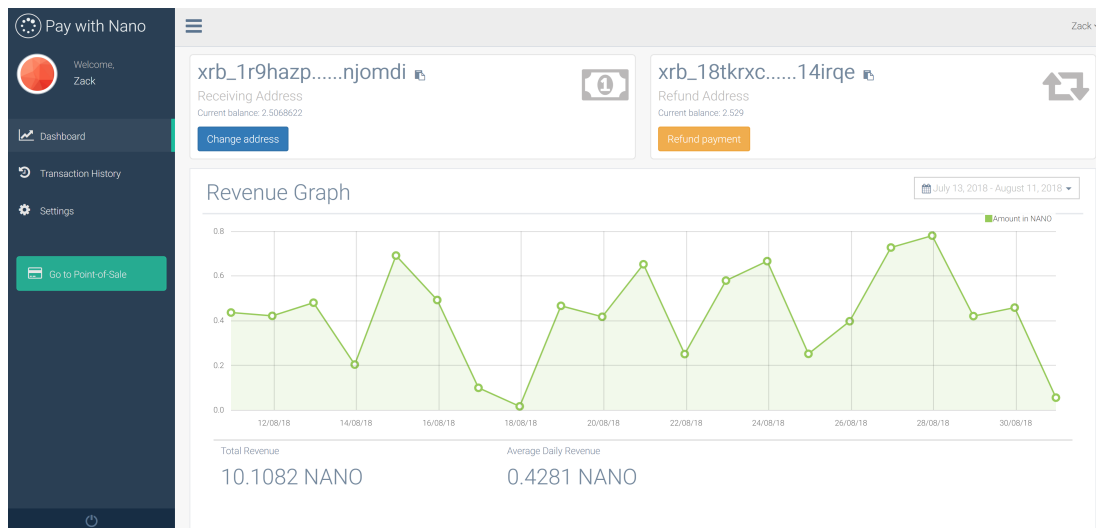


Figure 5.6: The home screen of the dashboard.

is then to minimise the potential consequences. Any action could be performed by either the merchant or the malicious customer, but only the merchant knows his own password. Therefore, password protection is added to all sensitive actions, which include changing the receiving address, checking the seed, and making a refund. Additionally, the system needs to prevent the malicious customer for initiating unauthorised payment requests, but requiring the merchant to input the password every time is both inconvenient and unsafe. Therefore, a 4-digit PIN is required instead, and the action of changing this PIN is password protected.

Another potential threat is a man-in-the-middle attack. The most relevant form of the man-in-the-middle attack, in this case, is the phishing attack, where the user is presented with a website that pretends to be this payment processor. If the user falls for the fake website, the attacker could then steal credentials or redirect payments to his own account. A graphical fingerprint representation, which has shown promising results, according to previous studies [45], is implemented to counter this form of attack. The idea behind this technique is to use a "fingerprint" of the user, a unique identification value, as the seed to generate an image. The user is then presented with the same image when interacting with the application. If the image ever changes, the user would know that something is wrong. Figure 5.7 shows the end result in this implementation.

In this case, the "fingerprint" used is the refund wallet identifier associated with the user. As mentioned in Section 2.2.2, this identifier is only stored on the local database, so the phishing site could not obtain it unless the database itself is

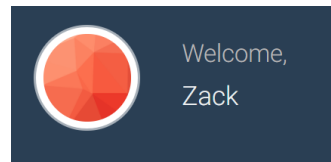


Figure 5.7: The graphical fingerprint representation is displayed beside the welcome text.

compromised. Since the image generation is done at the client side, the identifier is hashed before being used as the generation seed, masking the actual value from the client.

## 5.4 Responsiveness

Responsiveness is embedded in the design, allowing the user to accept payments on any device. It opens the application to service professionals, such as plumbers, who often work in the client's property and may have only a smartphone available. Figure 5.8 shows the consistency of the interface across three different devices.



Figure 5.8: The payment page presented on a smartphone (left), a tablet (middle), and a laptop (right).

The effort is focused on making the POS terminal the same experience across all types of devices. By contrast, the dashboard employs a desktop-first approach, where the experience is optimised for full-size screens. This is done under the assumption that the user would perform peripheral actions on larger-screen devices.

# Chapter 6

## Back-end Design

### 6.1 Information Delivery

To fulfil the requirements, the server needs to be exposed to different types of data as shown in Figure 6.1.

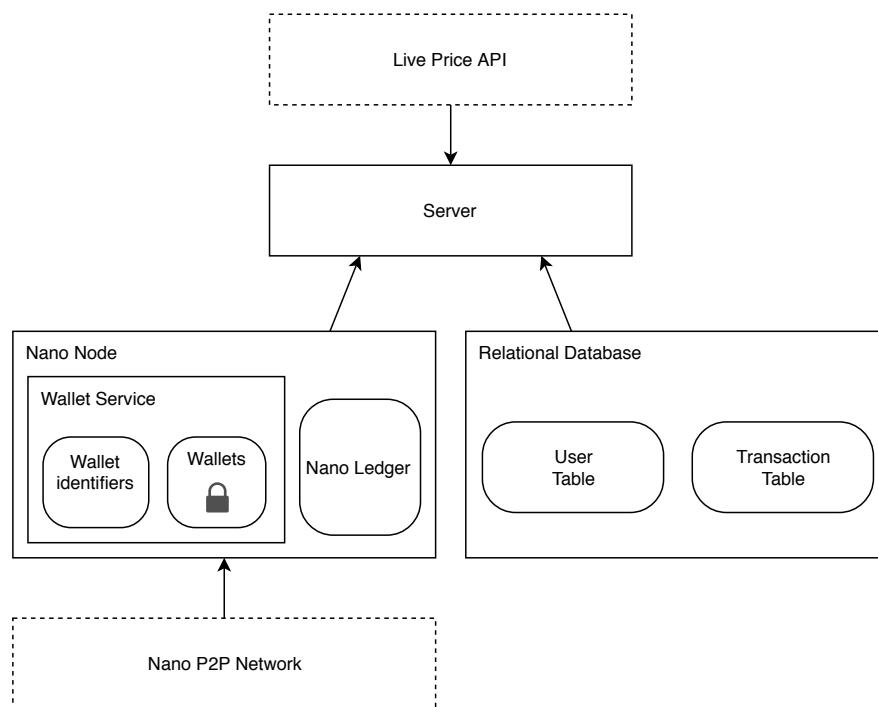


Figure 6.1: An overview of how the server obtains the required information. In this diagram, each rectangle represents a separate component. A rounded rectangle represents a persistent storage entity. Rectangles with dashed border are not under the project's responsibility.

The user and previous transaction data are stored in a relational database.

As previously discussed, the official Nano node contains a wallet management service. Thus, it is advantageous to let the node store the wallet data along with the Nano ledger. Each merchant is given a transition wallet and a refund wallet at registration. To retrieve either of the wallets, the system fetches the wallet identifier from the *User* table and uses it to locate the actual wallet data stored in the node. The communication between the server and the node is established through Remote Procedural Calls (RPCs).

## 6.2 Payment Flow

The high-level design of the flow of the payment is demonstrated in Figure 6.2

In essence, the payment processor uses the merchant's transition wallet to generate a new address for each payment. Note that the set of possible transition addresses for each merchant is disjointed to other merchants' set. The reason behind this design choice is further explained in Section 6.7. Once the payment arrives at the transition address, the system immediately forwards it to the merchant's receiving account.

At the first glance, it seems that introducing a layer of transition addresses between the customer and the merchant is redundant, but that is far from the truth. In the case where multiple payments of the same amount are expected, the system could use the layer to distinguish which customer has paid. The layer thus allows the merchant to operate multiple POS terminals by introducing separations between transactions. Furthermore, it could be used for handling under/overpayments by returning the corresponding amount back to the customer. The system could also forward a portion of the payment to multiple accounts. The service provider could charge a fee at this point. These changes are all feasible because of the flexibility added by the in-between layer.

This design minimises the time that it takes for the fund to become usable for the merchant. However, it also raises privacy concerns since the customer can trace forward to see all the payments that the merchant has received. This problem is difficult to address due to the transparent nature of Nano (and other non-privacy-centred cryptocurrencies). A possible solution could be to employ stealth addresses [46], but they are not a part of the Nano protocol, and no wallet provides support so far. Another possibility would be to allow the merchant to input multiple receiving addresses, and the system randomly chooses one when

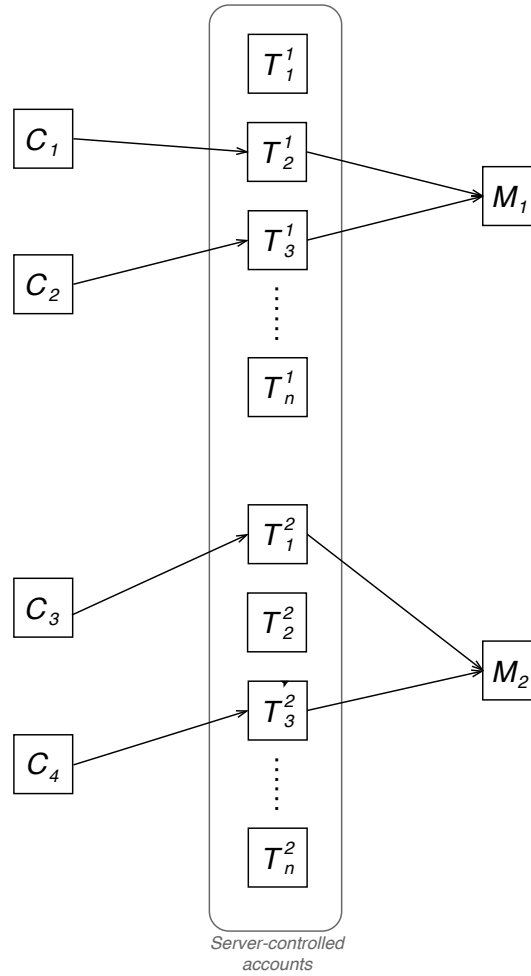


Figure 6.2: An example of the payment flow.  $C_i$  represents the account of the  $i^{th}$  customer.  $M_j$  represents the receiving account of the  $j^{th}$  merchant.  $T_l^k$  represents the  $l^{th}$  transition address associated with  $k^{th}$  merchant. Funds flow in the direction of the arrow.

forwarding a payment. However, this approach would drastically increase the set-up time for the merchant.

Ultimately, for the foreseeable future, the vast majority of the customers would be paying through other means in real-world settings, thus significantly reducing the severity of the privacy issue.

## 6.3 Payment Processor Design

Figure 6.3 describes the core components that would handle the payment processing logic.

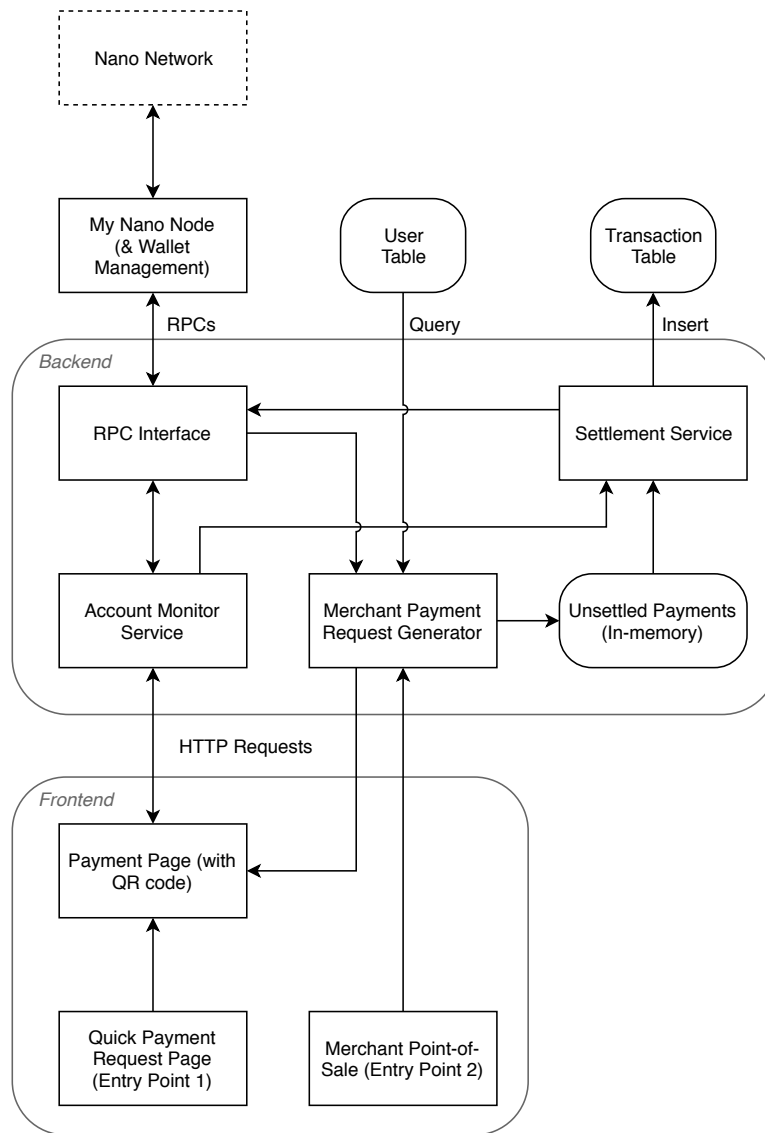


Figure 6.3: The architecture of the payment processing components. The arrows represent the direction of the information flow. A rounded rectangle represents a persistent storage entity. The peripheral components, such as the refund service and the authorisation service, are not included in the diagram.

As shown in the diagram above, there are two entry points for requesting payments. The first of which is the quick payment request page. It triggers the *Account Monitor Service* module to check whether a *send* block with the correct amount and address has been broadcasted. This serves as a simple feature for potential users to trial.

The exact method of detection employed by the *Account Monitor Service* module is an interesting discussion point. In general, this could be done either

through polling, which involves checking the state of the account at fixed intervals, or callbacks, which involves having the newly broadcasted blocks pushed to the server. Initially, it is thought that using callbacks would be the superior option since fewer resources are used. However, it is discovered that the node would occasionally fail to push a block. Although this event rarely happens, it does render the monitoring service unreliable.

A compromise is made in the final design. The main method of detection is still through callbacks, and an additional fallback mechanism is implemented where the account state is polled after relatively long intervals to pick up any missed relevant blocks.

The second entry point is the merchant POS page which utilises the core of the system. When the merchant initiates a payment request, the *Merchant Payment Request Generator* module retrieves the merchant's transition wallet and uses it to generate a transition address for the payment. The module then pushes the relevant information on to the *Unsettled Payments* hash table and redirects the view to the payment page. Of course, storing any information in-memory carries the risk of losing data due to unforeseeable circumstances such as unexpected server outages. The recovering process is explained in Section 6.7.

The *Settlement Service* module is triggered once the *Account Monitor Service* module has detected that the customer has transferred the required amount. It pops the corresponding entry from the *Unsettled Payments* dictionary and records this transaction onto the *Transaction* table. It then redirects the fund to the merchant's receiving address, in the exact manner described in the *Settlement Phase* shown in Figure 4.1.

## 6.4 Refund

Each merchant is given a refund account at registration, and it is up to the user to maintain a sufficient balance on the account. The application presents a list of refundable transactions where the merchant can choose from to initiate a refund. The system first checks that the original recipient is indeed the merchant, and the merchant's refund account contains sufficient balance before broadcasting the refund *send* block.

The amount of Nano in the refund is determined by picking the lower of two values: the original amount of Nano and the amount of Nano equivalent

to the original fiat value, based on the current exchange rate. This mechanism is introduced to prevent customers from using the price volatility to abuse the refund feature.

## 6.5 Security

As the system deals with the transfer of money, security becomes the main concern. This section discusses the security considerations that are present in the design.

The first line of defence comes from the system architecture. As previously mentioned, the wallet management responsibilities are delegated to the node. This means that the attacker has to access the node in order to obtain any sensitive wallet information. Such action is impossible without compromising the host machine, as the RPC port for the node is not exposed externally. The server is designed to run on the same machine as the node and talk to the node via RPCs locally.

Second, the system does not store passwords. The robustness of the wallet management service is discussed in Section 2.2.2. It is reasonable to assume that any fund in a refund wallet can only be accessed by unlocking it using the correct password. Thus, if the system does not store this information, the fund is still safe from the attacker in the event that the system is compromised. To achieve this design, the password given at registration is used to lock the user's refund wallet. When password verification is required, such as at login, the wallet management service runs the verification process, as described in Section 2.2.2, using the given password. Only during a refund should the wallet need to be actually unlocked. In which case, the system unlocks the wallet, broadcasts the refund *send* block, and immediately locks the wallet again.

## 6.6 Account Recovery

Of course, the consequence of the system ridding itself of passwords is that the user has the full responsibility of safekeeping it and the seed. If the user loses these, the fund on the refund account is unrecoverable, which again aligns with the full-ownership idea of cryptocurrencies.

If the user loses the password but still has the seed, he could recover his account by giving the seed to the application. The application could then use



the seed to generate the first deterministic address, which should be the original refund address. By matching this account to the database, the user is identified. The seed is then put into a new wallet, and the wallet is locked with a new password given by the user.

Note that if the first deterministic address does not match with anything on the record, the system would prompt the user to register. This effectively allows the user to create a new merchant's account using his own wallet for refunds.

## 6.7 System Recovery

Recalling Figure 4.1, once a payment has reached the layer of transition addresses, it is then the payment processor's responsibility to make sure that the payment is forwarded to the merchant's receiving address. During normal operations, forwarding funds are performed immediately when possible. However, if the system unexpectedly crashes as the customer is making a payment, the fund would be stuck in one of the transition addresses.

Therefore, a "sweep" routine is executed at the system initialisation stage. It iterates through all used addresses in all transition wallets and checks for funds, which could be either pending or claimed, depending on whether or not the node has the chance to broadcast the *receive* block before crashing. If any fund is found, the routine then performs the necessary steps to forward it to the corresponding receiving address. Since each wallet generates a disjoint set of addresses, the system could be absolutely certain of the rightful recipient.

However, as previously mentioned, due to the asynchronous nature of the Nano network, there is no notion of time. Therefore, it is impossible to determine the time of the original payment without querying an oracle (trusted third-party node). At the time of writing, there is no publicly-available API that can support this query.



# Chapter 7

## Implementation Details

### 7.1 Full Stack Overview

#### 7.1.1 Database

The foundation of every good application is a well-designed persistent storage layer. Accordingly, an SQLite relational database is chosen, as the tasks at hand are rigid and defined by straightforward relations. Two tables, namely the *User* table and the *Transaction* table, are created in the database.

The *User* table contains all the relevant information associated with each user. Table 7.1 details the fields that exist in the *User* table.

Table 7.1: The fields of the *User* table.

Field Name	Type (Size)	Unique?	Description
id	Integer	Yes	The unique primary key.
username	String (20)	Yes	The user's username for logging in.
email	String (32)	No	The user's email address
pin	String (4)	No	The 4-digit authorisation code for initiating a payment request.
refund_wallet_id	String (64)	Yes	It points to a locked refund wallet, used for refunding customer's payments and authorisation.
refund_address	String (64)	Yes	The first and the only address generated by the refund wallet
transition_wallet_id	String (64)	Yes	It points to a wallet for generating transition addresses.
receiving_address	String (64)	No	It points to an account, controlled by the merchant, for receiving payments.

The *Transaction* table contains all the relevant information for each transaction. Table 7.2 details the fields that exist in the *Transaction* table.

Note that the amount is stored as a string because there is a compatibility

Table 7.2: The fields of the *Transaction* table.

Field Name	Type (Size)	Unique?	Description
id	Integer	Yes	The unique primary key.
user_id	Integer	No	The id of the user who is receiving the payment.
timestamp	Datetime	No	Timestamp when the transaction is finalised.
source	String (64)	No	The customer's account.
destination	String (64)	No	The transition address for receiving this payment.
amount_nano	String (20)	No	The amount in Nano.
currency	String (4)	No	The based currency chosen, e.g. USD.
amount	String (20)	No	The amount in the base currency.
status	String (8)	No	Must be one of these values: "success", "timeout", "refunded".
hash	String (64)	Yes	The hash of the send block

issue with the *Decimal* type, and precision lost would incur if stored as a float.

As one of the tasks is to display the transaction history for any given user, a *receiving\_user\_id* is included for each transaction, allowing all transactions for any particular user to be fetched in one joint operation.

### 7.1.2 Nano Node

The Nano node is powered by the official distribution of the Nano software developed by the core team [47]. It supports all the RPC requests outlined in the RPC protocol [48]. At the time of writing, the newest version is v14.2.

### 7.1.3 Back-end

The back-end is built using Flask. In this application, Flask extensions, namely *Flask-Login*, *Flask-SQLAlchemy*, and *Flask-WTForms*, are included to support user authentication, data models, and form rendering, respectively.

With these modules installed, a model-view-controller (MVC) [49] architecture could then be established.

Flask supports a feature called "blueprints", which effectively allows separations within a project, where each smaller component groups a set of related controllers together. In this application, three blueprints are used to accommodate controllers for the dashboard, the payment pages, and the monitoring service,

Table 7.3: The supported endpoints and corresponding functions.

URL	Method	Parameters	Description
/	GET	-	Redirects to '/dashboard' if user is authenticated, else redirects to '/login'.
/login	GET	-	Renders the login page.
/login	POST	username, password	Redirects to '/dashboard' if username and password are verified.
/register	GET	-	Renders the register page.
/register	POST	username, password, email	Registers the new user. Redirects to '/login'.
/logout	GET	-	Logs out the current user. Redirects to '/login'.
/dashboard	GET	-	Renders the dashboard.
/history	GET	-	Renders the transaction history table.
/settings	GET	-	Renders the settings page.
/confirm_refund	GET	transaction_id	Checks if user can refund transaction_id. If valid, renders the confirm refund page.
/confirm_refund	POST	password	Performs the refund if password is verified.
/pay/	GET	currency, address, amount	Converts amount in currency to amount in Nano. Renders customer payment page and triggers 'api/payment_received'.
/pay/merchant	GET	-	Renders the POS terminal home page.
/pay/merchant	POST	currency, amount	Redirects to '/pay/', passing in currency, amount, and a newly generated transition address.
/api/payment_received	GET	address, amount	Monitors whether or not address receives amount, within a configured amount of time. Returns a transaction JSON.

respectively. The endpoints and the corresponding purpose are shown in Table 7.3.

The monitor service can be accessed by calling an asynchronous endpoint, where a response is only returned after the correct payment is detected or the transaction reaches a timeout.

The live exchange rates of Nano are provided by the CryptoCompare API [50]. The price is calculated using the Volume Weighted Average Price [51], which calculates a weighted average Nano price from seven reliable exchanges. The application supports USD, GBP, JPY, and EUR. Additional currencies can be easily incorporated, although the dropdown menu would no longer be the optimum design choice when there are many supported currencies.

#### 7.1.4 Front-end

A template engine replaces tokenised strings on a template with rendered strings, thus allowing web pages with customised information to be served to the user. Jinja2 [52] is the template engine that is incorporated into Flask. It supports simple logic such as if-statements and loops, as well as macros and base templates, allowing the final codebase to be "DRY" (don't repeat yourself).

The layout and the visual elements are borrowed and modified from a template called "Gentelella", which itself is based on the Twitter Bootstrap library [53]. The grid system of Bootstrap is used to adjust the appearance for different screen sizes and ultimately achieve responsiveness.

The asynchronous endpoint is consumed using a JavaScript promise [54], which dynamically updates the payment page to reflect the final outcome of the payment.

The graphical fingerprint representation is generated using a JavaScript library called "Trianglify.io".

## 7.2 Development Methodology

Since the knowledge about most of the technologies for this project is initially limited, the agile methodology is used for the development process to allow for adaptability.

A Kanban board [55] is set up for task organisation. This style of development focuses on finishing each task as fast as possible. Therefore, three tasks, at most,

are allowed to progress at any time, and any new task is restricted from starting until a progressing task is completed. The completed task log also serves as a good record for the report writing stage.

The project is generally divided into two development stages, corresponding to the two entry points described in Section 6.3. The completion of the first entry point establishes a working end-to-end flow of the components. The remaining features are extended from this foundation to facilitate the second entry point.





# Chapter 8

## Evaluation

The performance and the usability of the application are evaluated. The hardware used for the studies are as follows:

- **Payment processor host machine:** A Dell Windows 10 laptop equipped with an Intel i7-6700HQ CPU.
- **Customer wallet:** Nano Wallet app for Android by Nano Wallet Company [56], a popular Nano wallet implementation, running on a Samsung Android smartphone.
- **Merchant wallet:** Same set up as the customer wallet.

### 8.1 Performance Evaluation

#### 8.1.1 Payment Time

The average time required for the completion of one payment cycle is evaluated. In this simple scenario, the payment processor is only handling one payment at any time. To simulate a normal checkout, the Nano Wallet app is used to facilitate the payment from the smartphone. The timer starts when the payment appears to have been sent on the Nano Wallet app and stops when the merchant's receiving account, which is another account generated by the Nano Wallet app, receives and claims the payment. Recalling Figure 4.1, the customer only needs to wait until the *payment phase* finishes, and this interval is also recorded. This experiment is logistically difficult to automate because it is dependent on the

interaction between the two devices. It is eventually decided to send 100 test payments manually and sequentially. The corresponding time is recorded.

Initially, the experiment is planned to be conducted by running another node on a different machine for sending payments. After synchronising the clocks, scripts could be run on both machines to achieve automation. However, it is not clear whether this node would be a sensible replacement to the node for the Nano Wallet app, which means that the simulation might not be a true reflection of the actual real-world scenario. Therefore, the manual approach is used instead.

The median time required to complete the *payment phase* is found to be 9.6 seconds. The corresponding histogram is shown in Figure 8.1. The median time required to complete the *settlement phase* is found to be 28.2 seconds. The corresponding histogram is shown in Figure 8.2. It takes a median time of 38.7 seconds for the whole payment sequence to complete. The corresponding histogram is shown in Figure 8.3. From the users' perspective, the typical wait is just under 10 seconds, which is acceptable but not ideal.

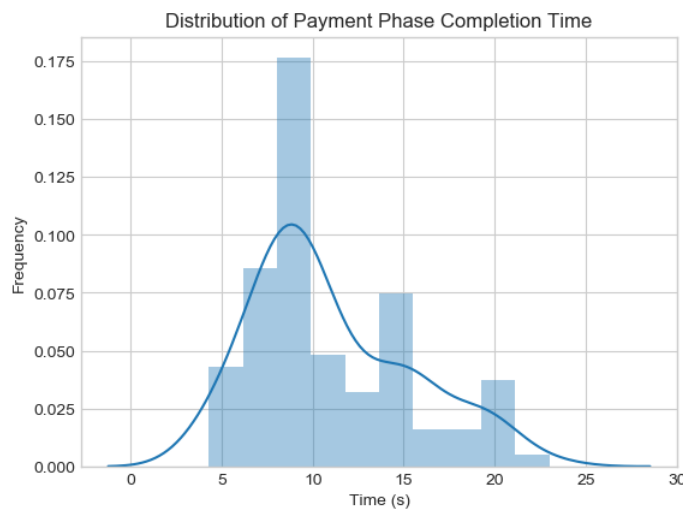


Figure 8.1: Histogram of the completion time of the payment phase, gathered from 100 runs.

During the experiment, it is discovered that the node sometimes consumes almost all the bandwidth available, and the payment acknowledgement is significantly slower than periods of lower bandwidth usage. Therefore, it is believed that the waiting time could be reduced by hosting the node on a cloud server. The large bandwidth usage issue is primarily due to the inefficiency during rebroad-

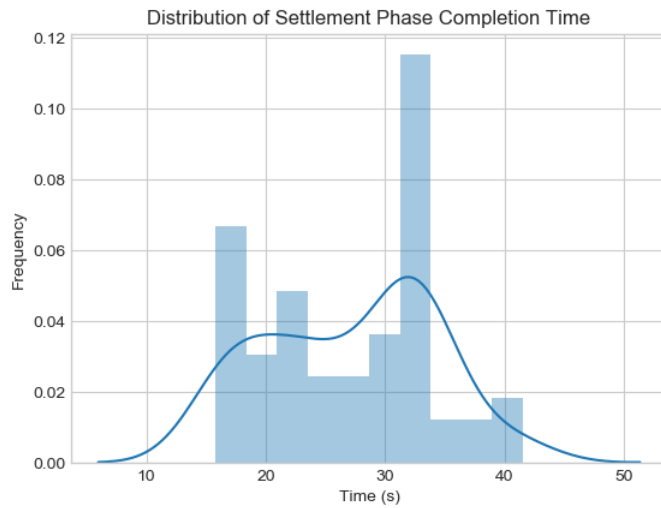


Figure 8.2: Histogram of the completion time of the settlement phase, gathered from 100 runs.

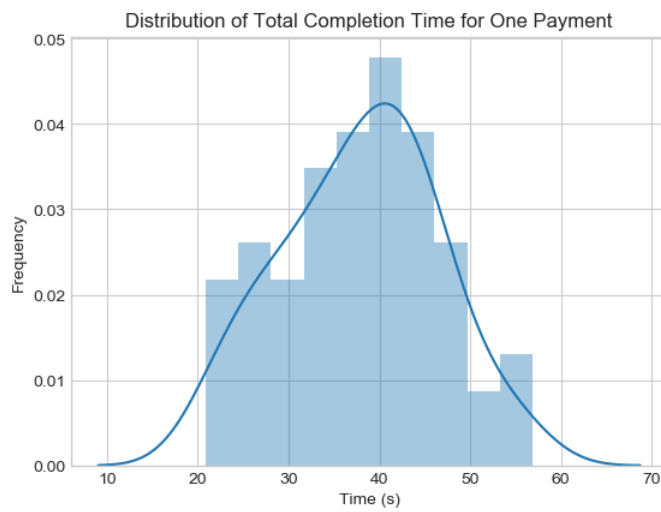


Figure 8.3: Histogram of the total time spent for one transaction cycle, gathered from 100 runs.

casting and voting. At the time of writing, an updated protocol that reduces the bandwidth usage by 80% is being implemented by the Nano development team [57].

### 8.1.2 Recovery Testing

A script is written to test the "sweep" routine that runs when the web server initialises. Ten test merchant accounts are created, and their associated transition wallets are obtained. Ten addresses are generated from each wallet, and a random small amount of fund is sent to each address. Some of the funds are not initially claimed.

The web server is then initialised, and the "sweep" routine is executed. After the completion of the routine, the final receiving account balances are verified. This script is executed ten times, and the correct balances are observed after every run. It is worth noting that the "sweep" routine, in every run, took around 15 minutes to complete. Most of this time is used to generate PoW for the transactions.

## 8.2 Usability Study

In this case, usability is defined as the capacity for an average university student to use the payment processor application to accomplish common tasks without major errors. A Think Aloud study is conducted to evaluate the usability. The methodology is influenced by a previous usability study [58].

### 8.2.1 Participant Selection

The ideal focus group would be the shopkeepers, but they are difficult to recruit. Given the time constraints, five students are selected instead. They are considered a group large enough to identify principal usability issues, yet small enough to analyse the qualitative data effectively.

Within the student demographics, it is ensured that the participants are not biased towards a particular subgroup. This presumption is achieved by selecting participants with different degree subjects and different levels of understanding about cryptocurrencies. This information is collected in a short questionnaire (see Appendix D). The results of which are presented in Table 8.1, demonstrating the diversity of backgrounds.

Table 8.1: Participant background. Knowledge towards cryptocurrency is determined by the answer to the questionnaire. It ranges from 0 to 4, where 0 represents strongly disagree with the statement and 4 represents strongly agree with the statement.

Participant No	Age	Gender	Degree Level	Course Subject	Knowledge towards Cryptocurrency
1	25	Female	Masters	Philosophy	1
2	27	Female	Masters	High Performance Computing	0
3	23	Male	Masters	Accounting & Finance	3
4	24	Female	Masters	Architecture	1
5	24	Male	Masters	Informatics	4

### 8.2.2 Methodology

Each participant is first given a consent form for permission of recording and data use (see Appendix C). The participant is then instructed that he will be using a cryptocurrency payment processor. He is instructed to ‘talk aloud’ as he solves the tasks and is reminded that he should talk constantly, verbalising what he is seeing and thinking as he interacts with the application.

The participant is given a practice problem to familiarise himself with the process. He is told that his questions would not be answered while completing the task, as this could affect his behaviour and responses towards the application.

The participant is presented with three tasks (see Appendix B) and the interface. The screen capturing and the voice recording then follows. The participant proceeds to attempt to complete the tasks. Short notes are taken during the process to mark points of interest. See Appendix A for the full researcher script.

When all the sessions have been completed, the notes are reviewed, and interesting timestamps are revisited in the recording to further examine the participant’s thoughts and behaviours. For each participant, a list of issues is identified, which highlights the points at which they struggled or did not take the expected path to completion. These lists are combined to produce a full record of issues. As Think Aloud is a qualitative evaluation method, all issues are treated with equal importance, regardless of the frequency of occurrence.

### 8.2.3 Results

From the investigation, the following significant issues are identified:

1. At the POS terminal, the "Base Currency" drop-down menu is sometimes ignored. The participants proceed to request 0.50 Nano instead of £0.50.
2. When completing the details, the participants are often confused and distracted by the "Changing Receiving Address" form, and they would proceed to fill the form with invalid details, before realising their mistake.
3. After completing task 2, the participants are often unsure about how to return to the dashboard.
4. After identifying the correct transaction to refund, the participants tend to click on the "View" button first, which takes them to a confusing external Nano block explorer.

### 8.2.4 Solutions

The following changes are proposed to resolve the issues identified:

- Merge the "Base Currency" drop-down menu into the "Amount" textbox. Change the texts to symbols, e.g. change "GBP" to "£". This should resolve issue 1.
- Dynamically update a conspicuous text area with the amount in both Nano and the selected currency as the user is typing the amount. This should help solve issue 1.
- In forms, add a help text that would pop-up beside the input textbox when the textbox is in focus. This should partially resolve issue 2.
- Add a short tutorial that presents to the user if the user is yet to complete all of the details. The tutorial should quickly explain the Nano cryptocurrency and how to set up a Nano wallet. This change should solve issue 2 and help solve issue 1.
- Add an "Exit" button onto the POS terminal. The terminal tab should close when the button is pressed. This change should solve issue 3.
- The "View" buttons in the transaction history table should redirect the user to a simple page highlighting only the most relevant pieces of information of the corresponding transaction, which are the timestamp, the amount, and the status. This change should resolve issue 4.

# Chapter 9

## Conclusion

In this project, *Pay with Nano*, a cryptocurrency payment processor for the cryptocurrency Nano is successfully designed and implemented. Its performance is assessed by performing test payments that simulate real-world settings. It is found that it takes a median time of 9.6 seconds to confirm a customer payment and then a median time of 28.2 seconds for the merchant to receive the payment. It is thought that both figures could be further reduced. The application's usability is evaluated by conducting a Think Aloud study.

The final product meets the specifications outlined in Section 4.1. In particular:

- The implemented payment process matches with that described in 4.1
- The application supports inputting amount in multiple currencies and dynamically converts it into the equivalent amount in Nano, using a live-price API.
- The application contains a dashboard that displays statistics and the transaction history. The transaction history can be downloaded as a CSV file or printed.
- A refund account is allocated to each merchant account. The merchant could top up the refund account and refund previous transactions.
- Responsiveness is achieved by utilising Twitter Bootstrap's grid system.
- Simultaneous payments are enabled by introducing the transition account layer.

- Security and minimum trust are ensured. The system prevents external attackers from stealing a user's money by simply not storing the user's password and the plain seed. It prevents malicious customers from performing unintended actions by attaching a security check to every sensitive operation.
- Ease-of-use is achieved by introducing familiarity and simplicity through careful user interface design.

## 9.1 Critical Analysis

Overall, the project is completed according to the plan. However, at the initial stages, the Django web application framework is used. It is then discovered that Django supports many features irrelevant for this project, which consequently creates unnecessary complexity. Therefore, the framework is swapped for Flask, but a lot of time is lost in learning and developing with Django.

The Think Aloud usability study yields insightful results, but the changes are difficult to implement as the front-end and the back-end are already connected. The study should be carried out before the front-end styling starts. The participants could be given mock-ups in the Think Aloud session.

Nano is an interesting cryptocurrency of choice. Working with it reveals that it is not yet a fully-mature technology and does require more work, especially with the unreliable RPC callback feature. However, the protocol is being actively developed, with exciting improvements being introduced all the time, starting with the drastic reduction of bandwidth usage. It is believed that Nano has the potential to become a top cryptocurrency, as the eventual technological goal for blockchain is to facilitate fee-less and instantaneous value transfer. At the time of writing, Nano is the only working product that reaches this goal.

## 9.2 Future Work

The project currently allows the merchant to receive Nano as payment, which can be considered as ideal if the merchant speculates that the price of Nano is going to increase in the future and holds the received Nano as a form of investment. However, if the merchant is apprehensive about the price volatility of Nano or



cryptocurrencies in general, he has no better option but to manually convert it into a stable asset. Therefore, a natural extension of this project is to automate this process, and this is where Nano's fast settlement time stands out. It is theoretically possible to convert the payment from Nano to a stablecoin [59], such as Tether (USDT), in less than one minute from when the customer has paid, utilising the 5-second deposit time on the Binance exchange [60]. The payment would be settled directly into the Nano address for the merchant's Binance account, and a script would be immediately triggered to call the Binance API and perform the conversion, for a fee of no more than 0.2% of the converted value [61]. This service does, however, require the user to entrust it with his API key. As a result, keeping this key safe would then become the main security concern of the system.

For additional future work, the user experience could be further improved by reducing the waiting time that both the merchant and the customer have to endure at checkout. As previously mentioned, this improvement could be achieved by deploying the application, along with the node, onto a cloud server.



# Appendix A

## Think Aloud Researcher Script

Hello, my name is Zack. Today we will be using a cryptocurrency payment processor web app based on a cryptocurrency called Nano. I will ask you to perform some tasks using the app.

Your participation today is purely voluntary, and you may stop at any time. The purpose of this exercise is to evaluate the design of this app. Please remember I am testing the design. I am not testing you.

In this observation, I am interested in what you think about as you perform the tasks I am asking you to do. In order to do this, I am going to ask you to talk aloud as you work on the task. What I mean by “talk aloud” is that I want you to tell me everything you are thinking from the first time you see the statement of the task till you finish the task. If you feel that you are unable to fully complete a task, please state that you cannot complete the task.

I do not want you to plan out what you say or try to explain to me what you are saying. Just act as if you are alone, speaking to yourself. It is most important that you keep talking. If you are silent for a long period of time, I will ask you to talk. Do you understand what I want you to do? Good. Now I will begin with some practice problems. First, I will demonstrate by thinking aloud while I solve a simple problem: “How many meals will I have until the end of this week?” [Demonstrate thinking aloud.]

Now it is your turn. Please think aloud as you work out how many windows are in your mother’s home. [Let them finish]

Good. Now, those problems were solved all in our heads. However, when you are using the app, you will also be looking for things, and seeing things that catch your attention. These things that you are searching for and things that you see

are as important for my observation as the thoughts you are thinking. So please verbalize these too. As you are doing the tasks, I won't be able to answer any questions. But if you do have questions, go ahead and ask them anyway so I can learn more about what kinds of questions the app brings up. I will answer any questions after the session. Also, if you forget to think aloud, I'll say, "please keep talking." Do you have any questions about the think aloud?

Now I have some tasks printed out for you. When carrying out these tasks, please imagine you are a merchant sitting at the checkout table of your shop. I am going to be the customer and perform a payment with my phone. I will prompt you when I do so.

Please be aware from this point onwards I will be recording your voice and capturing your interaction with the app. [Hand them the tasks.]

Here are the tasks you will be working on. Please read all the tasks aloud so you can get comfortable with speaking your thoughts. Do you have any questions about the tasks? Great. You may begin.

# Appendix B

## Think Aloud Tasks

### Task 1

Log in using the following credentials:

*Username: Zack*

*Password: 1111*

Complete any additional set up if required. Launch the Point-of-Sale terminal.

### Task 2

A customer comes with a purchase of £0.50. Initiate a payment request and proceed when you believe that the payment has completed.

### Task 3

The customer changed his mind and returned the purchased item. Please refund the customer.



# Appendix C

## Think Aloud Consent Form

This research aims to gather information about the design of a cryptocurrency payment processor. Today I will be asking you to carry out tasks using the app.

If you allow it, I will be audio recording the interview. If you feel uncomfortable about this at any time you may stop the recording or tell me that the next bit should not be quoted. Recordings will be used in my research to learn about the design.

The audio will be deleted after the transcripts are made. The transcripts will be kept for a maximum of one year and then destroyed. Anonymized quotes from the transcripts may be included in the research report.

The research is supervised by Dr Christophe Dubach (christophe.dubach@ed.ac.uk) and conducted by Zack Zixu Xiang (s1408280@sms.ed.ac.uk).

- ☐ I understand that I am participating in a study as part of the evaluation of the design of the prototype.
- ☐ I am willing to be digitally recorded and transcribed for use as part of the research project.
- ☐ The researcher may use quotes from this study in publications provided that the quote is anonymised and cannot be connected back to me.

Participant Name: \_\_\_\_\_

Date: \_\_\_\_\_

Participant Signature: \_\_\_\_\_

Researcher Name: Zack Zixu Xiang

Date: \_\_\_\_\_





# Appendix D

## Think Aloud Questionnaire

Name: \_\_\_\_\_

Age: \_\_\_\_\_

Gender: \_\_\_\_\_

Degree level (currently pursuing): \_\_\_\_\_

Course name or subject: \_\_\_\_\_

Please read the following statement and tick the appropriate circle:

“I have considerable knowledge of cryptocurrencies.”

Strong Agree

☐

Agree

☐

Neutral

☐

Disagree

☐

Strongly Disagree

☐



# Bibliography

- [1] E. Central Bank, “Payment statistics for 2016,” 2017. [Online]. Available: <https://www.ecb.europa.eu/press/pdf/pis/pis2016.pdf?be9989f6bd72483ebe27d8dfae1f0362>
- [2] W. Jevons, “Money and the mechanism of exchange,” p. 352, 1859.
- [3] “Fiat Money and Gold | Sunshine Profits.” [Online]. Available: <https://www.sunshineprofits.com/gold-silver/dictionary/gold-fiat-money/>
- [4] “Is the Fed’s Inflation Target Kaput? - Bloomberg.” [Online]. Available: <https://www.bloomberg.com/news/articles/2018-02-02/is-the-fed-s-inflation-target-kaput>
- [5] “Probability-impact assessment - Praxis Framework.” [Online]. Available: <https://www.praxisframework.org/en/library/probability-impact-assessment>
- [6] S. H. Hanke and A. K. F. Kwok, “On the measurement of Zimbabwe’s hyper-inflation,” *Cato J.*, vol. 29, p. 353, 2009.
- [7] “Venezuela: inflation could top 1 million percent by year’s end, IMF warns | World news | The Guardian.” [Online]. Available: <https://www.theguardian.com/world/2018/jul/24/venezuela-inflation-million-percent-imf-warning>
- [8] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System.” [Online]. Available: [www.bitcoin.org](http://www.bitcoin.org)
- [9] “Bitcoin USD - Yahoo Finance.” [Online]. Available: <https://finance.yahoo.com/quote/BTC-USD/chart>
- [10] “Block Size And Transactions Per Second | BitcoinPlus.org.” [Online]. Available: <https://www.bitcoinplus.org/blog/block-size-and-transactions-second>
- [11] “Big transaction fees are a problem for bitcoin.” [Online]. Available: <https://www.cnbc.com/2017/12/19/big-transactions-fees-are-a-problem-for-bitcoin.html>
- [12] “Small Business Retail | Visa.” [Online]. Available: <https://usa.visa.com/run-your-business/small-business-tools/retail.html>

- [13] C. Pérez-Soi, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, “Double-spending Prevention for Bitcoin zero-confirmation transactions.” [Online]. Available: <https://eprint.iacr.org/2017/394.pdf>
- [14] “Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa.” [Online]. Available: <https://powercompare.co.uk/bitcoin/>
- [15] “Bitcoin Energy Consumption Index - Digiconomist.” [Online]. Available: <https://digiconomist.net/bitcoin-energy-consumption>
- [16] “UK | Worldpay.” [Online]. Available: <https://www.worldpay.com/global/about/regional-expertise/uk>
- [17] “Visa Outage: Switch Failure Blamed, No Detection Software in Place.” [Online]. Available: <https://www.cbronline.com/news/visa-outage>
- [18] “Nano – an instant, zero-fee, scalable currency.” [Online]. Available: <https://nano.org/en>
- [19] C. Lemahieu and C. Co, “Nano: A Feeless Distributed Cryptocurrency Network,” Tech. Rep., 2014. [Online]. Available: <https://nano.org/en/whitepaper>
- [20] A. Back, “Hashcash-A Denial of Service Counter-Measure,” Tech. Rep., 2002. [Online]. Available: <http://www.hashcash.org/papers/hashcash.pdf>
- [21] Z. Xiang, “Informatics Project Proposal,” The University of Edinburgh, Tech. Rep., 2018.
- [22] “Ledger Pruning | Nano Developers.” [Online]. Available: <https://developers.nano.org/roadmap/ledger-pruning/>
- [23] “Universal Blocks Specification · nanocurrency/raiblocks Wiki.” [Online]. Available: <https://github.com/nanocurrency/raiblocks/wiki/Universal-Blocks-Specification>
- [24] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.
- [25] S. Popov, “The Tangle,” Tech. Rep., 2017. [Online]. Available: [http://iotatoken.com/IOTA\\_Whitepaper.pdf](http://iotatoken.com/IOTA_Whitepaper.pdf)
- [26] “Timelock - Bitcoin Wiki.” [Online]. Available: <https://en.bitcoin.it/wiki/Timelock>
- [27] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An overview of blockchain technology: Architecture, consensus, and future trends,” in *Big Data (Big-Data Congress), 2017 IEEE International Congress on*. IEEE, 2017, pp. 557–564.

- [28] “Nanode · Representatives.” [Online]. Available: <https://www.nanode.co/representatives>
- [29] “Rebroadcasting-Elections.md at master · NanoTools/wiki.” [Online]. Available: <https://github.com/NanoTools/wiki/blob/master/Rebroadcasting-Elections.md>
- [30] “RaiBlocks Deterministic Account Addresses – Coinmonks – Medium.” [Online]. Available: <https://medium.com/coinmonks/raiblocks-deterministic-keys-8cb869cc6046>
- [31] “BLAKE2.” [Online]. Available: <https://blake2.net/>
- [32] “ED25519.” [Online]. Available: <https://ed25519.cr.yp.to/>
- [33] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.
- [34] “Object-relational Mappers (ORMs) - Full Stack Python.” [Online]. Available: <https://www.fullstackpython.com/object-relational-mappers-orms.html>
- [35] “Flask (A Python Microframework).” [Online]. Available: <http://flask.pocoo.org/>
- [36] “The Web framework for perfectionists with deadlines | Django.” [Online]. Available: <https://www.djangoproject.com/>
- [37] “Thinking Aloud: The #1 Usability Tool.” [Online]. Available: <https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/>
- [38] E. Charters, “The use of think-aloud methods in qualitative research an introduction to think-aloud methods,” *Brock Education Journal*, vol. 12, no. 2, pp. 68–82, 2003.
- [39] “How Square Works | Square.” [Online]. Available: <https://squareup.com/gb/en/2018/how-square-works>
- [40] “BitPay.” [Online]. Available: <https://bitpay.com/>
- [41] “Coinbase Commerce.” [Online]. Available: <https://commerce.coinbase.com/>
- [42] The UK Cards Association, “UK Card Payments Summary,” Tech. Rep., 2017.
- [43] “Account Registration design pattern example at Lessaccounting - 13 of 206.” [Online]. Available: <http://ui-patterns.com/patterns/AccountRegistration/examples/882>
- [44] “What is Defensive Design? - Simplicable.” [Online]. Available: <https://simplicable.com/new/defensive-design>
- [45] J. Tan, L. Bauer, J. Bonneau, L. F. Cranor, J. Thomas, and B. Ur, “Can Unicorns Help Users Compare Crypto Key Fingerprints?” 2017. [Online]. Available: <http://dx.doi.org/10.1145/3025453.3025733>

- [46] “Stealth Address | Moneropedia | Monero - secure, private, untraceable.” [Online]. Available: <https://getmonero.org/resources/moneropedia/stealthaddress.html>
- [47] “Releases · nanocurrency/raiblocks.” [Online]. Available: <https://github.com/nanocurrency/raiblocks/releases/>
- [48] “Nanocurrency RPC Protocol,” p. <https://github.com/nanocurrency/raiblocks/wiki/RPC>. [Online]. Available: <https://github.com/nanocurrency/raiblocks/wiki/RPC-protocol>
- [49] “MVC architecture - App Center | MDN.” [Online]. Available: [https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern\\_web\\_app\\_architecture/MVC\\_architecture](https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture)
- [50] “CryptoCompare API.” [Online]. Available: <https://www.cryptocompare.com/api>
- [51] “Volume Weighted Average Price (VWAP).” [Online]. Available: <https://www.investopedia.com/terms/v/vwap.asp>
- [52] “Welcome | Jinja2 (The Python Template Engine).” [Online]. Available: <http://jinja.pocoo.org/>
- [53] “Bootstrap · The world’s most popular mobile-first and responsive front-end framework.” [Online]. Available: <https://getbootstrap.com/docs/3.3/>
- [54] “JavaScript Promises: an Introduction | Web Fundamentals | Google Developers.” [Online]. Available: <https://developers.google.com/web/fundamentals/primers/promises>
- [55] “What is a Kanban Board? | LeanKit.” [Online]. Available: <https://leankit.com/learn/kanban/kanban-board/>
- [56] “Nano Wallet Company, LLC | The best way to manage your Nano.” [Online]. Available: <https://nanowalletcompany.com/>
- [57] “Pull Request #1025 · nanocurrency/raiblocks.” [Online]. Available: <https://github.com/nanocurrency/raiblocks/pull/1025>
- [58] Z. Xiang and H. Bacon, “Evaluation of Usability of Smart Refrigerator App Design,” Tech. Rep., 2017.
- [59] “Explaining Stable Coins, The Holy Grail Of Cryptocurrency.” [Online]. Available: <https://www.forbes.com/sites/shermanlee/2018/03/12/explaining-stable-coins-the-holy-grail-of-cryptocurrency>
- [60] “Binance Deposit Speed Comparison | Anything Crypto.” [Online]. Available: <https://www.anythingcrypto.com/fastest-exchange-deposit/binance>
- [61] “Binance - Fee Schedule.” [Online]. Available: <https://www.binance.com/en/fee/schedule>