

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

A Government-Backed Cryptocurrency for Universal Basic Income

Author:
Matthew S. Morrison

Supervisor:
Prof. William J. Knottenbelt

Submitted in partial fulfillment of the requirements for the MSc degree in
Computing Science of Imperial College London

September 2018

Abstract

As Artificial Intelligence increasingly threatens human employment globally, the concept of Universal Basic Income, in which a government provides all citizens a fixed, regular payment regardless of income, is increasingly being suggested as a solution.

Yet governments, and indeed many recent UBI trials, have so far relied on traditional financial infrastructure to process these types of transactions. We believe that the use of cryptocurrency and blockchain technology can transform the process, making it cost effective, flexible and, most importantly, transparent.

The cryptocurrency-based UBI systems that exist today tend to lack real financial value, with the supply of their tokens growing *ad infinitum* and thus exhibiting deflationary mechanics. Furthermore, these systems tend to lack a robust identification process (with many using SMS-based or social media sign up mechanisms), meaning that they are susceptible to Sybil attacks, in which users can sign up for multiple accounts.

In this project we have designed and implemented a government-backed UBI system using smart contracts on the Ethereum blockchain. In order to give the token real financial value, businesses can exchange their received UBI tokens for real Ether at a pre-determined exchange rate. This underpins the value of the token, giving businesses an incentive to accept the token as payment for their goods and services.

In order to prevent Sybil attacks, this project implements a system of zero-knowledge proofs using a modified version of the ZoKrates software. This system allows users to prove that they hold key information on themselves (existing government data including name, national insurance number, date of birth and address), without disclosing this information to the entire public blockchain during the sign up process. The only parameters visible to the network after sign up will be eight elliptic curve pairings and the public inputs (the first 32 bits of the SHA256 hash).

Our React-based web application allows consumers to sign up to the service (hiding the underlying complexity of the zero-knowledge proofs) and spend their UBI tokens at registered businesses. Furthermore, functionality within the smart contracts allows third parties to use the UBI token in their applications.

UBI experts have endorsed the key elements of the project, including the use of zero-knowledge proofs for user identification and backing the token with real value. Whilst a number of improvements can be made, including the reduction of costs and a second layer of identification, the consensus is that this project has developed an effective solution for a government-backed Universal Basic Income.

Acknowledgments

I am grateful to my supervisor, Prof. William J. Knottenbelt, for his enthusiasm and help throughout the duration of my project and studies.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Aims and Objectives	2
1.3	Contributions	3
1.4	Outline	7
1.5	Legal, Ethical and Social Considerations	8
1.6	Statement of Originality and Publications	8
2	An Overview of UBI	9
2.1	The History of UBI	9
2.2	Arguments For UBI	11
2.3	Arguments Against UBI	14
2.4	The Growing Need for UBI	15
2.5	Non-Cryptocurrency UBI Experiments	16
2.6	Cryptocurrency UBI Solutions	17
2.7	Wörglcoin	18
3	Smart Contract Design	19
3.1	An Overview of Ethereum	19
3.2	Smart Contract Programming	20
3.3	Functionality	21
3.4	Security	25
3.5	Optimisation	30
3.6	Testing	32
4	Identification	34
4.1	The Identification Problem	34
4.2	Existing Solutions	35
4.3	Interactive Zero-Knowledge Proofs	37
4.4	Non-Interactive Zero-Knowledge Proofs	39
4.5	Zk-SNARKs in Ethereum	51
4.6	WörglCoin’s Identification System	56
5	Using Wörglcoin	58
5.1	Functionality Overview	58
5.2	UBI Token Interface	69

6	Evaluation	70
6.1	Evaluation Methodology	70
6.2	Interviews	71
6.3	User Feedback	75
6.4	Merchant Feedback	75
6.5	Summarising the Feedback	81
6.6	Addressing the Feedback	82
7	Conclusion	83
7.1	Summary of Achievements	83
7.2	Evaluation Overview	84
7.3	Applications	85
7.4	Future Work	86
	Bibliography	88
	Glossary	93
	Appendices	95
A	Ethics Checklist	95
B	Legal, Social and Ethical Considerations	98
C	Non-Cryptocurrency UBI Experiments	99
D	Cryptocurrency UBI Solutions	100

List of Figures

1.1	Elon Musk advocates the introduction of UBI	2
1.2	Mark Zuckerberg advocates the introduction of UBI	2
1.3	The UBI system design	3
1.4	Gas costs of the Wörglcoin smart contract	4
1.5	WörglCoin’s identification system	5
1.6	The web application homepage	6
2.1	Key historical figures	10
2.2	England’s poor laws in 1834	10
2.3	UK Government spending on welfare in 2016-2017	11
2.4	Inequality rates over time	13
2.5	Labour share of national income	13
2.6	Cost of UBI scheme in the UK	14
2.7	Potential rates of job automation by country	15
3.1	The consumer structure used in the WörglCoin smart contract.	21
3.2	The business structure used in the WörglCoin smart contract	22
3.3	The typical structure used to keep track of key elements	22
3.4	Modifiers ensure functions are being executed as intended	23
3.5	Events trigger updates to the user interface	24
3.6	Measures taken to prevent a ‘re-entrancy’ attack	26
3.7	OpenZeppelin’s safe arithmetic operations library	26
3.8	Gas costs of the Wörglcoin smart contract	31
3.9	Smart contract unit testing	32
3.10	Smart contract code coverage	33
4.1	The Ali Baba cave	37
4.2	An R1CS representation of the flattened code	41
4.3	Step 1 of the QAP transformation	43
4.4	Step 2 of the QAP transformation	44
4.5	Step 3 of the QAP transformation	45
4.6	Step 4 of the QAP transformation	46
4.7	The QAP problem in terms of polynomials	46
4.8	The solution to the polynomial QAP problem	47
4.9	An elliptic curve	48
4.10	Point representation of an elliptic curve	49

4.11 WörglCoin's identification system	57
5.1 The final design of the Wörglcoin system	58
5.2 Contract owner user guide: step one	60
5.3 Contract owner user guide: step two	60
5.4 Contract owner user guide: step three	61
5.5 Contract owner user guide: step four	61
5.6 Business user guide: step one	62
5.7 Business user guide: step two	63
5.8 Business user guide: step three	63
5.9 Business user guide: step four	64
5.10 Business user guide: step five	64
5.11 Consumer user guide: step one	66
5.12 Consumer user guide: step two	66
5.13 Consumer user guide: step three	67
5.14 Consumer user guide: step four	67
5.15 Consumer user guide: step five	68
5.16 Consumer user guide: step six	68
5.17 The Wörglcoin smart contract interface.	69
6.1 Arguments for UBI from Alex Howlett	73
6.2 Arguments for UBI from Otto Lehto	73
6.3 Arguments against UBI from Donald Hirsch	73
6.4 Key features of UBI from Otto Lehto	73
6.5 Blockchain is not currently ready from Lennard Hulsbos	74
6.6 Advantages of identification method from Alex Howlett	74
6.7 Criticism of identification method from Alex Howlett	74
6.8 Advantages of devaluation mechanism from Alex Howlett	74
6.9 Google survey given to users	76
6.10 Question one of the user survey	77
6.11 Question two of the user survey	77
6.12 Question three of the user survey	78
6.13 Question four of the user survey	78
6.14 Question five of the user survey	79
6.15 Question six of the user survey	79
6.16 Question seven of the user survey	80
6.17 Question eight of the user survey	80

Chapter 1

Introduction

Key technologists have advocated the introduction of Universal Basic Income as a policy to mitigate the threat of Artificial Intelligence. In this project we have designed and implemented a government-backed UBI system using smart contracts on the Ethereum blockchain. In order to prevent Sybil attacks, a system of zero-knowledge proofs is in place, allowing unique individuals to sign up to the scheme without disclosing any private information.

1.1 Motivation

With Artificial Intelligence (‘AI’) threatening to displace human workers across many different industries, economists and technologists have been grappling with potentially dire consequences of such a transition, including higher levels of unemployment and growing wealth inequality. One solution that has been put forward by leading technologists, including Elon Musk [1] (see Figure 1.1) and Mark Zuckerberg [2] (see Figure 1.2), is Universal Basic Income (‘UBI’). UBI is a system in which a government (or private entity) gives all citizens a fixed, regular payment regardless of the citizen’s income level, age or employment status.

In 2017, we saw many countries and technology firms experimenting with UBI. The majority of these systems propose direct payments to citizens, much like many welfare payments are distributed today. This project explores the potential use of cryptocurrency and blockchain technology to implement UBI. Such a system would allow a low-fee way to transfer UBI funds (no middlemen), offer transparency (all payment mechanisms are implemented in code) and provide flexibility (in case parameters of the scheme need to be changed).

An area of focus will be the drawbacks of current cryptocurrency-backed UBI schemes, including the limited value of a typical UBI token (acting as a disincentive to businesses to sell their goods) and the problem of uniquely identifying citizens to receive the token. This project adds to the literature on UBI schemes by exploring and implementing the technology using an Ether-backed UBI coin and zero-knowledge proofs to privately identify citizens.

“I think there is a pretty good chance we end up with Universal Basic Income... due to automation. I am not sure what else one would do. People will have time to more interesting, more complex things. We have to figure how to integrate in the future with advanced A.I., that's going to be one of the toughest challenges.” [1]

Figure 1.1: Elon Musk advocates the introduction of UBI

“Today, we have a level of wealth inequality that hurts everyone... It is time for our generation to define a new type of social contract... We should explore ideas like Universal Basic Income to make sure everyone has a cushion to try new ideas.” [2]

Figure 1.2: Mark Zuckerberg advocates the introduction of UBI

1.2 Aims and Objectives

The aims and objectives of this project are to:

- Examine existing implementations of UBI using traditional direct cash payments and cryptocurrencies.
- Develop a new Ethereum-based UBI token which is backed by Ether to give it real financial value.
- Thoroughly test the security and efficiency of the smart contracts underpinning the UBI token.
- Implement a robust and secure zero-knowledge proof based identification system.
- Develop a web application to allow users an easy way to sign up to the scheme and spend their tokens on real products.

1.3 Contributions

This project designs and implements an Ether-backed UBI token with a zero-knowledge proof based system for citizen identification. The specific contributions are outlined below.

Implementation of an Ether-backed UBI coin

A key drawback of current UBI systems using cryptocurrency is that the tokens tend to lack real financial value. Given that these schemes create a token in which supply will grow *ad infinitum*, businesses often have little incentive to offer their products in exchange for a deflationary UBI token. This leads to users having very little choice regarding the items they can purchase with their tokens.

In this project an Ether-backed UBI token has been created to solve this problem. The system has been built using smart contracts (see Figure 1.3), allowing the contract owner to transfer Ether into the contract and set an Ether value for tokens. Once a business has received UBI tokens for their products, this will be converted to real Ether at a pre-determined exchange rate. This underpins the value of the token, giving businesses an incentive to accept the token as payment for their goods and services.

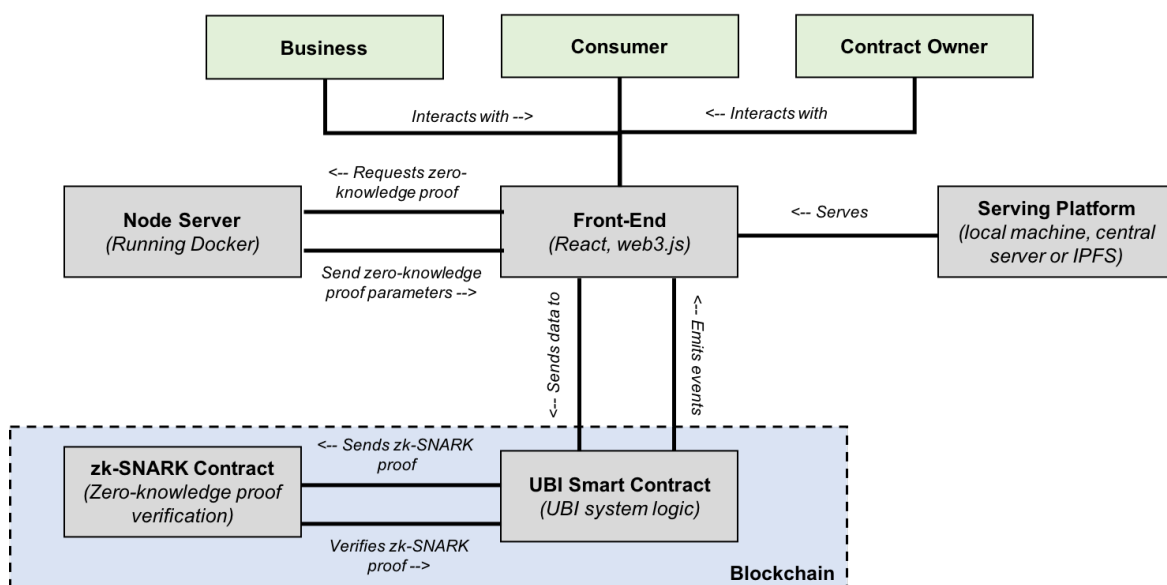


Figure 1.3: The UBI system design

In-Depth Smart Contract Security and Efficiency Review

One of the key benefits of implementing a UBI system using cryptocurrency and blockchain technology is cost. Blockchain technology allows rules and parameters to be written in code and easily changed. Furthermore, given the lack of middlemen in the operation (e.g. individuals administering payments), the cost of a cryptocurrency-based UBI system is relatively low.

For this reason, this project places a lot of focus on smart contract efficiency. There are two main costs when it comes to deploying smart contracts: the one-off initial contract deployment and the ongoing cost of function execution. A thorough review of smart contract efficiency has been conducted (see Figure 1.4), with an outline of all the steps taken to reduce the cost of implementing the UBI system.

Furthermore, the reputation of blockchain applications has suffered in recent years from several high-profile security exploitations. This can often occur from small mistakes in the smart contract code, or a lack of understanding around key functionality of a smart contract.

Therefore, this project conducts an in-depth security review of the smart contracts, outlining the key vulnerabilities typically seen and the steps that have been taken to mitigate these vulnerabilities. As a result, the UBI system implemented in this project is relatively low cost and secure, both critical in a government system of this kind.

Gas Price (Gwei)	3
Gas Price (Ether)	0.000000003
USD Per Ether	228.12
Gas Price (USD)	0.00000068436

Function	Gas Units	Cost Bearer	Cost (Gwei)	Cost (Ether)	Cost (USD)
Contract Creation + Deployment	266,666,667	Government	800,000,000	0.80	182.50
addBusiness	159,465	Government	478,395	0.00	0.11
addConsumerHash	44,755	Government	134,265	0.00	0.03
buyItem	189,970	Consumer	569,910	0.00	0.13
changeOwner	28,882	Government	86,646	0.00	0.02
changeTokenValue	28,176	Government	84,528	0.00	0.02
consumerSignUp	3,142,076	Consumer	9,426,228	0.01	2.15
makeComplaint	67,351	Consumer	202,053	0.00	0.05
markOrderAsSent	66,824	Business	200,472	0.00	0.05
changeTokenBalance	28,270	Government	84,810	0.00	0.02
resetComplaint	28,146	Consumer	84,438	0.00	0.02
resetTokenBalance	31,139	Government	93,417	0.00	0.02
sellItem	291,114	Business	873,342	0.00	0.20
topUpContract	43,181	Government	129,543	0.00	0.03

Figure 1.4: Gas costs of the Wörglcoin smart contract

Zero-Knowledge Proof Based Identification System

One of the most challenging aspects of a UBI system is identifying unique individuals to receive your coin. Without a robust system of identification, the scheme is vulnerable to Sybil attacks, where malicious users create fraudulent accounts in order to subvert and impede the functionality of your system.

Given that the government already holds a significant amount of pre-existing data on citizens (e.g. National Insurance numbers, resident address and age), this project explores the idea of users supplying their own information to prove they are indeed a unique citizen.

However, given that the public, permissionless Ethereum blockchain has been used for the implementation, no transaction can be private. This means that a user would not be able to submit details to the smart contract for verification without broadcasting their private details to every participant on the network.

This project implements a system of zero-knowledge proofs (see Figure 1.5), allowing users to prove they hold key information on themselves, without disclosing the information to the entire network.

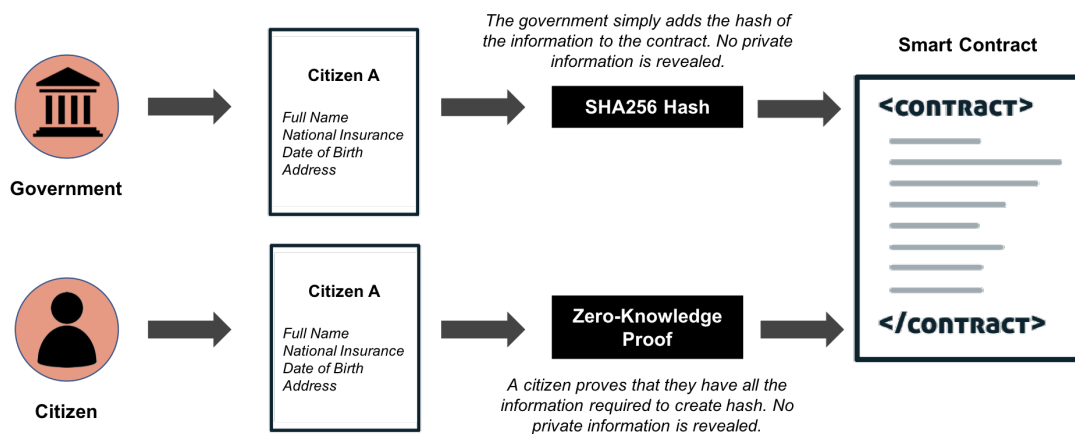


Figure 1.5: WörglCoin's identification system

Web Application for Sign Up and Spending

Our React-based web application (see Figure 1.6) allows consumers to sign up to the service and spend their UBI tokens at registered businesses. The web application also allows businesses to supply new products, and track all orders that have been made.

Furthermore, functionality within our system of smart contracts allows third party applications to accept the UBI token on their website. Their applications just need to query and execute certain functions, allowing them to check a user's balance, buy an item and transfer tokens.

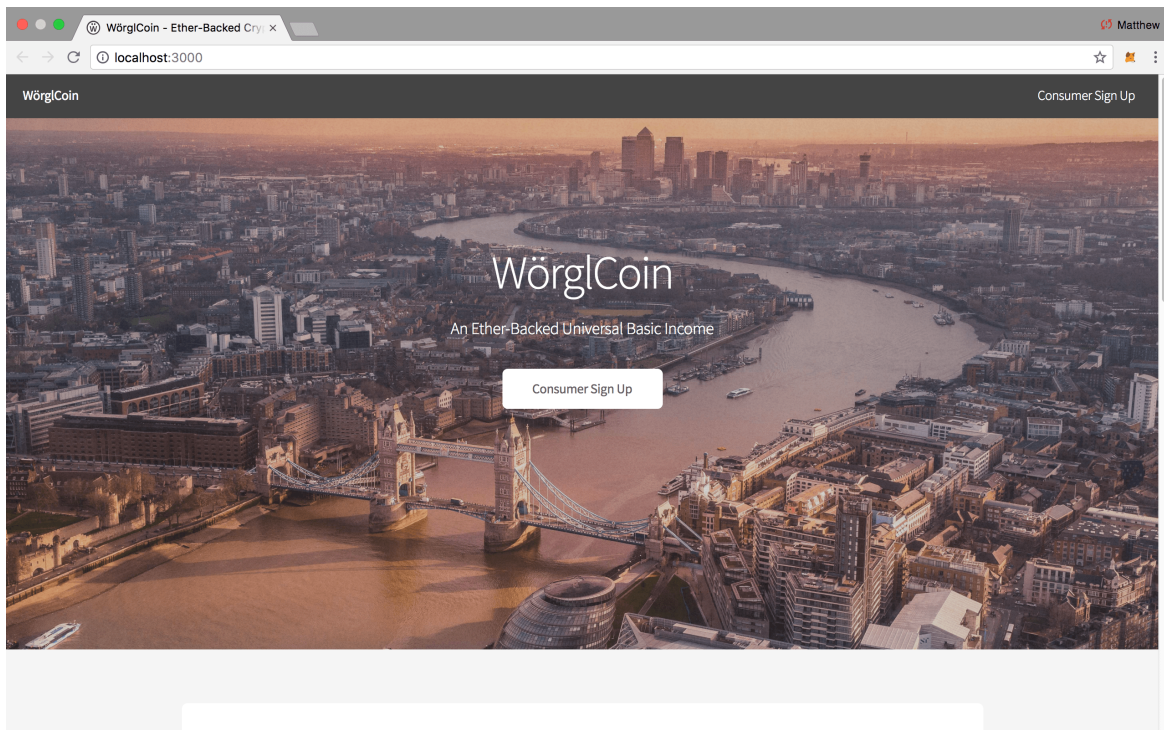


Figure 1.6: The web application homepage

1.4 Outline

The remainder of this report is organised as follows:

Chapter 2 examines the concept of Universal Basic Income, the arguments for and against such a scheme and the real-world implementation of schemes worldwide. Furthermore, a new UBI token is introduced, WörglCoin, which is backed by Ether and has a mechanism incentivising users to spend the token.

Chapter 3 outlines the design and implementation of the UBI system using Ethereum smart contracts. The main functionality of the system is examined, allowing consumers and businesses to sign up and exchange tokens for goods and services. Furthermore, a thorough review of security is conducted, looking into potential security vulnerabilities and measures that have been implemented to mitigate these vulnerabilities. A number of methods have been used to minimise the cost of the system, both for the one-off deployment of the smart contracts and ongoing function execution.

Chapter 4 examines the problem of uniquely identifying individuals in a UBI system. A number of solutions have been implemented by existing cryptocurrency-based UBI schemes and their limitations are discussed. A thorough overview of zero-knowledge proofs is conducted, exploring the mathematics behind them, as well as how they are implemented on Ethereum today. The UBI system outlined in this project uses pre-existing government data, alongside zero-knowledge proofs, to ensure that citizens are uniquely identified.

Chapter 5 outlines the implementation and functionality of the React-based web application, allowing consumers to sign up to the service and spend their UBI tokens at registered businesses. This chapter also examines the interface that has been created, allowing third party applications to accept the UBI token on their sites.

Chapter 6 outlines the evaluation process of the project. This has been conducted through interviews with Universal Basic Income experts and user testing with potential UBI recipients and merchants.

Chapter 7 concludes the report by summarising the achievements of this project and highlighting the potential opportunities for further work.

1.5 Legal, Ethical and Social Considerations

Consideration of all legal, ethical and social considerations has been made. A detailed table outlining all legal, ethical and social considerations can be found in Appendix A. A detailed explanation of all considerations can be found in Appendix B.

1.6 Statement of Originality and Publications

I declare that this thesis was composed by myself, and that the work that it presents is my own except where otherwise stated.

Chapter 2

An Overview of UBI

This chapter covers the economic and technical material necessary to understand the rationale and implementation of a UBI system. Furthermore, the use of cryptocurrency in creating a UBI system is explored and justified.

2.1 The History of UBI

The concept of a minimum basic income was first being discussed at the beginning of the 16th century by fellow humanists Thomas More [3] and Johannes Ludovicus Vives [4] (see Figure 2.1 [5] [6]). They both argued that such a scheme would improve the quality of an individual's life and prevent them from resorting to crime. This line of argument inspired many subsequent schemes, including a system of poor relief entitled 'England's Poor Laws' [7] (see Figure 2.2 [8]).

Much of the following academic thinking on the topic concentrated on the equal ownership of land as the foundation of a minimum standard of living. For example, Joseph Charlier [9] proposed granting every citizen a fixed payment on the basis of the rental value of all real estate.

Universal Basic Income ('UBI') as a concept was reignited in the 20th century by a number of economists, including Milton Friedman (although he referred to it being implemented as a negative income tax) [10]. Furthermore, Martin Luther King Jr. was a strong advocate of a guaranteed minimum income to provide equality and the abolition of poverty [11].

The modern interpretation of UBI (and the one we shall be using throughout this report) is the government provision of a fixed payment to all citizens, regardless of income level, employment status or any other discriminating factor. The only stipulations of the payment should be that the individual is a citizen of that country.



Thomas More



Johannes Ludovicus Vives

Figure 2.1: Key historical figures



Figure 2.2: England's poor laws in 1834

2.2 Arguments For UBI

Argument 1: Cut Welfare Costs

A common argument in support of UBI is that it could dramatically reduce government bureaucracy. By providing a single payment to every citizen, this could replace the hugely complex and convoluted welfare system many Western economies have in place today.

In 2016-2017, the UK government spent c.£217bn on the welfare state [12] (equivalent to c.£8,000 per household), including c.£92bn on the state pension and c.£27bn in personal tax credits (see Figure 2.3). Whilst the cost of implementation is hard to estimate, it can confidently be said that the system employs a huge number of people with many separate disparate systems.

Implementing just one payment to each citizen, without any bias or means-testing, could potentially eliminate a huge amount of overhead in the welfare system.

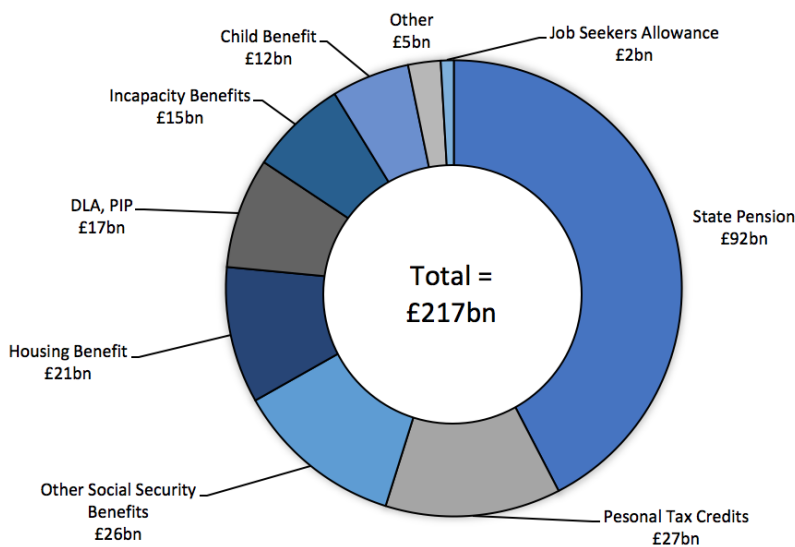


Figure 2.3: UK Government spending on welfare in 2016-2017

Argument 2: End Welfare Trap

The second common argument is that UBI is necessary to end the welfare trap (also called the ‘unemployment trap’). Individuals who rely on means-tested welfare payments are often disincentivised from rejoining the labour force, as their absolute income sometimes falls as a result of finding work. This distorts the incentives within our economy and can move people away from seeking productive work.

Giving all citizens a fixed monthly payment, which does not change as a result of a change in employment status or income level, guarantees that working always pays. There will be no perverse opportunity cost from rejoining the labour force and economic incentives within our economy will be fully aligned.

Argument 3: Boost Employment in Undervalued Sectors

Another argument for UBI is to boost employment within sectors that are traditionally under-rewarded by the free market. Allowing everyone a basic standard of living regardless of their job means that individuals will be able to pursue more socially ‘important’ jobs (such as charity work and community care) without sacrificing their quality of life.

Argument 4: Justice and Equality

Today, the top 10% of earners account for c.37% of national income in Europe, c.47% in North America and c.61% in the Middle East. Since 1980, income inequality has greatly increased across the globe [13], as seen in Figure 2.4 [13, p. 10].

Income inequality can be largely attributed to unequal ownership of capital within an economy, whether that be physical or intellectual capital. Since the 1990s, productivity gains driven by technology have been broadly captured by the owners of capital, as demonstrated by the dramatic reduction in the labour share of income [14], as seen in Figure 2.5 [14, p. 122].

Arguably, new technological innovations such as Artificial Intelligence will reduce the labour share of income either further, as many routine-based jobs are automated, with firms paying creators of the technology instead of their own labour.

UBI has the potential to recoup some of these lost funds, either through taxation or other schemes, and redistribute them to members of society. As a result, the impact of technological innovation on inequality will be dampened.

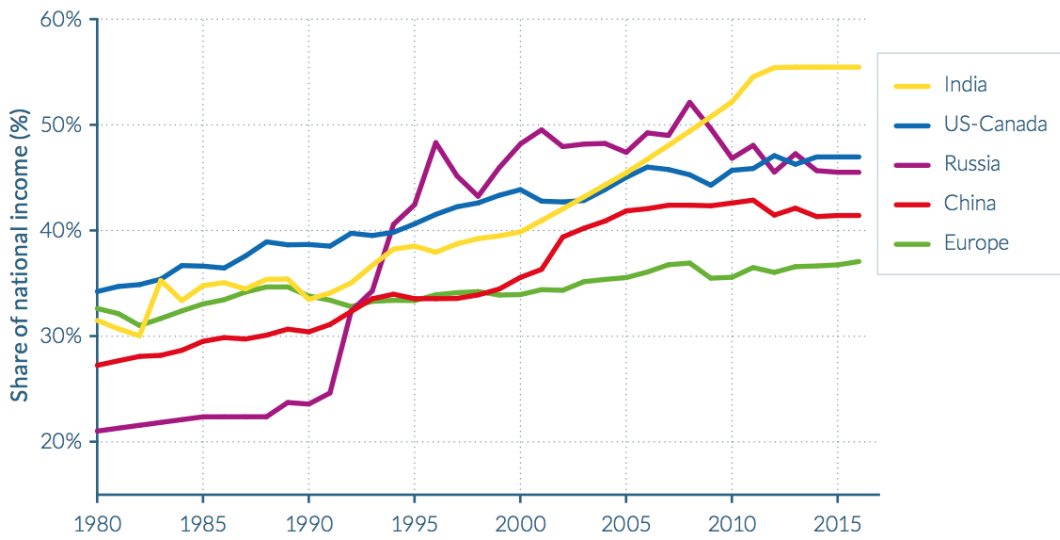


Figure 2.4: Inequality rates over time

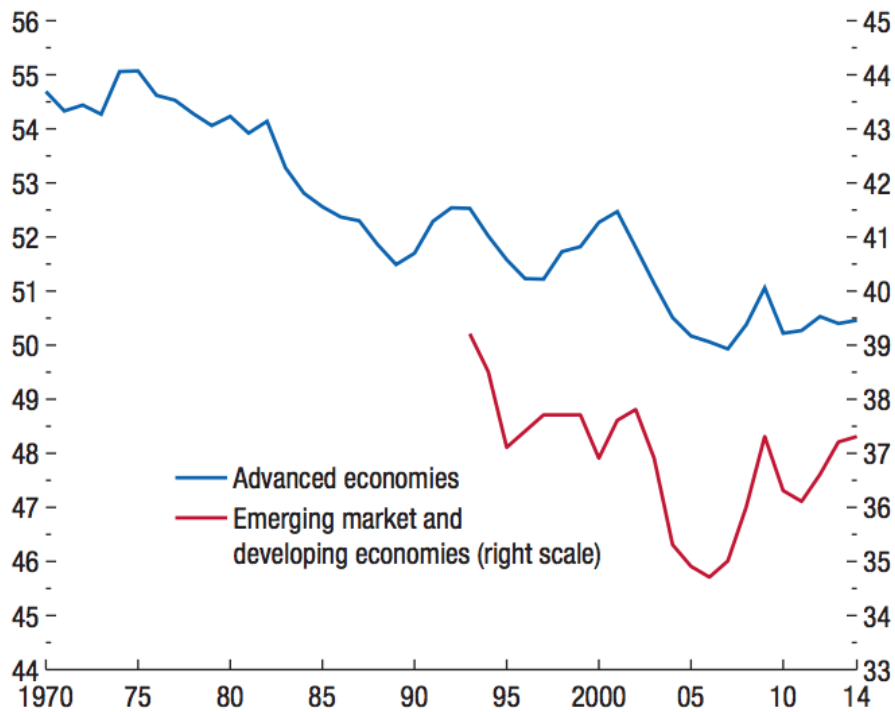


Figure 2.5: Labour share of national income

2.3 Arguments Against UBI

Argument 1: UBI Would Be Too Costly

One of the principle arguments against the implementation of UBI is that it would be far too costly. If every citizen in the UK above the age of 16 were to receive a fixed payment of c.£10,000 per year, this would cost a total c.£540bn (see Figure 2.6).

This is far greater than the current welfare budget of c.£217bn discussed earlier in this chapter. Therefore, this cost would have to be met by an increase in taxes, reduction in government spending or some other revenue generating venture.

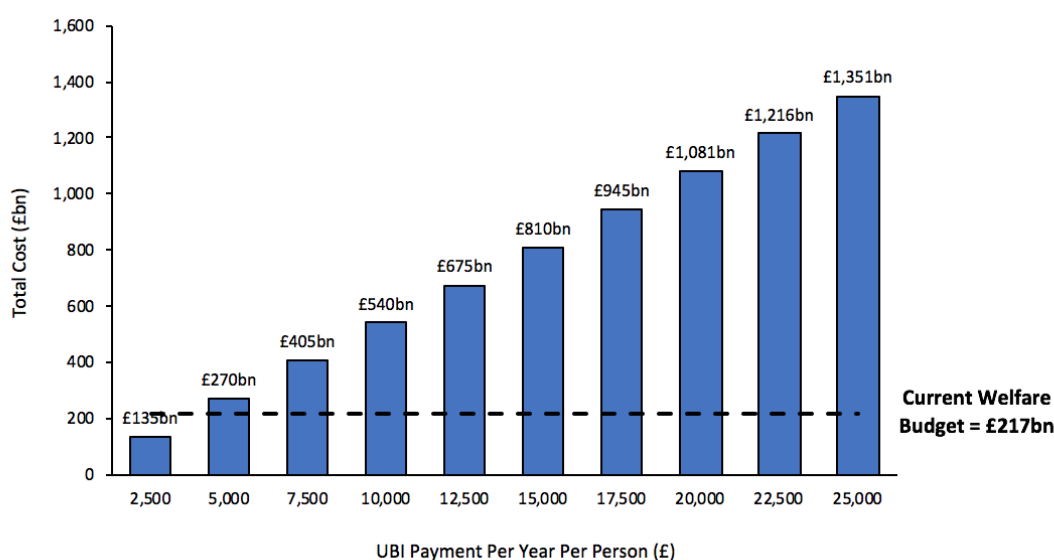


Figure 2.6: Cost of UBI scheme in the UK

Argument 2: Employment Disincentive

The second key argument against UBI is that it would act as a disincentive for individuals to participate in the labour force. The argument is that if individuals are allowed a comfortable life without the need for work, people will simply choose this option. By removing the need to work, the state is simply encouraging idleness.

Argument 3: Untested and Risky

Another argument against the implementation of UBI is that it is an untested idea, whereas the welfare system is a long-standing institution that, arguably, works. By implementing UBI on a mass scale, society takes a huge risk.

2.4 The Growing Need for UBI

According to a recent study by PwC, about 30% of current jobs in the UK are at risk of automation by 2030 [15] (see Figure 2.7 [15, p. 10]), with this figure rising to 45% in the manufacturing industry. The UK is not isolated, this will be a global trend, with close to 40% of jobs in the USA at high risk of automation.

Many Silicon Valley entrepreneurs, including Elon Musk [1], have suggested that UBI is going to be necessary in years to come. Bill Gates has even gone as far as suggesting that a ‘robot tax’ could be implemented to fund such a scheme [16].

Given that technological innovations such as Artificial Intelligence are likely to concentrate even more wealth in the hands of the few (only a handful of companies possess A.I. capabilities), exacerbating inequality across the globe, UBI is becoming an increasingly popular policy.

In Europe, a 2017 study found that that 68% of 11,000 Europeans would vote for a basic income referendum if one were immediately held in their country [17]. The concept has been at the forefront of Politics in the UK in recent months, with the Shadow Chancellor, John McDonnell, arguing for the widespread implementation of Universal Basic Income [18]. However, to date, no country has introduced a fully-deployed UBI scheme in any form.

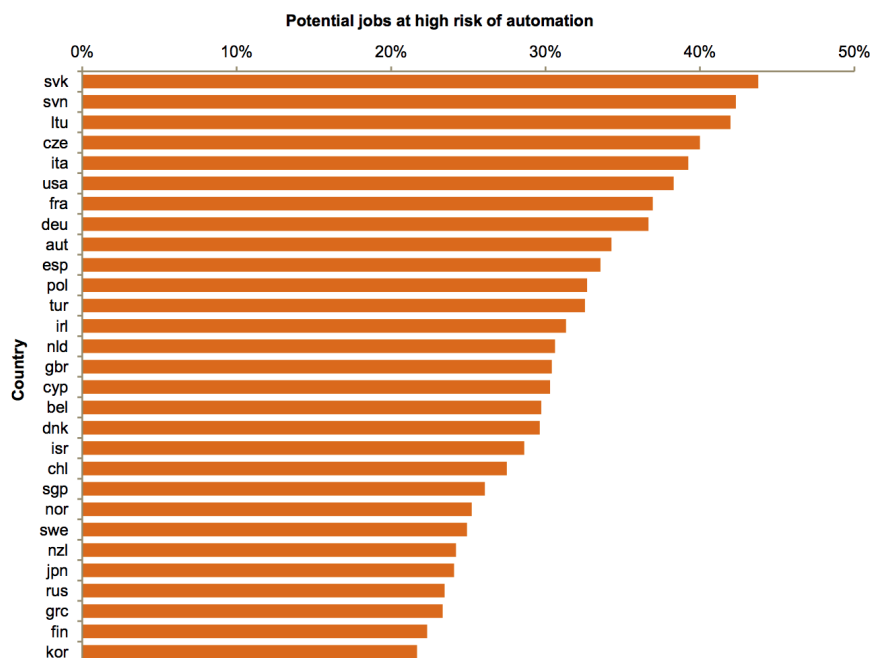


Figure 2.7: Potential rates of job automation by country

2.5 Non-Cryptocurrency UBI Experiments

There have been many recent examples of experimental UBI schemes that do not utilise cryptocurrency. A table summarising all of these experiments can be found in Appendix C. Two of these experiments that are implementing a ‘full’ UBI scheme (unconditional, significant value cash transfers) will be explored in depth.

Government-Backed: Finland

In 2017 and 2018, Finland’s Social Insurance Institution, Kela, implemented an experimental UBI scheme [19]. As part of the scheme, 2,000 participants between the ages of 25 and 58 were selected at random (although they were selected from those receiving basic unemployment allowance) in December 2016.

The participants are being paid €580 per month for a period of two years (1 January 2017 to 31 December 2018). The payment is paid unconditionally: it is not means tested and is not reduced by any other supplemental income the participant may have (in case they find employment). The results of the study are expected to come out in 2020 and all of the comparisons will be made to a control group of participants who did not receive a UBI payment.

Privately-Backed: Y Combinator

The Silicon Valley Seed Accelerator, Y Combinator, began experimenting with a privately-funded UBI scheme in 2017 [20]. As part of the experiment, c.3,000 participants across two US states have been recruited. Around 1,000 of these participants will receive US\$1,000 per month for a period of three to five years.

Y Combinator will obtain a vast amount of quantitative and qualitative data as part of the experiment, including details related to how participants use their time and how the UBI payment affects their children and people in their network.

2.6 Cryptocurrency UBI Solutions

The use of cryptocurrencies is not new in the UBI space (a summary of all solutions can be found in Appendix D). The use of a cryptocurrency can allow a flexible, efficient, and cost effective way of implementing UBI. In general, there are a number of fundamental problems with existing cryptocurrency UBI solutions.

The most fundamental flaw in these systems tends to be a lack of real currency backing for the token. Given you are distributing tokens at no cost, this inflates the supply of this token and thus it devalues. In such a system it is hard to incentivise producers to sell items for your token.

The second fundamental problem that these solutions have is identification. It can be difficult to ensure that there is a one-to-one mapping between an individual and each wallet. A number of schemes have been put in place to ensure that individuals do not set up multiple beneficiary UBI wallets.

Mannabase

Originally called Grantcoin, Mannabase [21] was created in May 2015 and is the first cryptocurrency to be managed and distributed by a tax-exempt US charity. Mannabase is the platform that allows users to sign-up and receive the Manna UBI, as well as acting as a social network allowing users to interact and send tokens to one another.

Manna is one of the most widely known and used UBI schemes, with over 3,500 holders of either Grantcoin or Manna at the beginning of 2018. As of the beginning of June 2018, the Manna coin had a market capitalisation of just over US\$200,000, having reached a peak of c.US\$2,000,000 at the end of 2017.

Manna basic income is distributed weekly into web-based wallets on the Mannabase platform and users can receive bonuses for signing up others onto the platform. In order to be eligible to receive Manna, a user has to go through a sign up process, entering details such as their address and mobile phone number. The user then has to enter a unique code sent to the mobile number specified.

There are fundamental problems with this implementation. Firstly, no real financial backing to the token has led to an extremely small market capitalisation. Secondly, deflationary economics has played a part and there are not many items that you can purchase using your Manna UBI tokens.

2.7 Wörglcoin

The inspiration behind this project is the Wörgl experiment that took place in Austria between 1932 and 1933. In this experiment, the small town of Wörgl in Austria experimented with implementing its own currency during the midst of the Great Depression.

The mayor of Wörgl at the time, Michael Unterguggenberger, faced a tough situation: the city had 1,500 unemployed individuals and a further 200 families had no money whatsoever. There were a large number of jobs that needed to be done in the town and an adequate number of labourers to do these jobs.

The problem was that he only had 40,000 schillings in the bank - nowhere near enough to pay for all the work. This is a situation an economy can regularly find itself in - plenty of free labour to do the job but not enough liquid currency to ensure everyone is willing to do the job.

Michael Unterguggenberger decided to create his own currency. Instead of spending the 40,000 schillings, he decided to deposit the money in a savings account and use it to back Wörgl's own 40,000 schillings worth of stamp scrip. A stamp needed to be applied to the currency in order to retain its value (i.e. a negative carry on the currency, incentivising people to spend), at a rate of 1% each month.

The result of this implementation was to significantly increase the velocity of currency, more people were put into work, and ultimately all required projects were completed. Cryptocurrency can be defined in such a way to have incentivisation mechanisms built-in, unlike traditional fiat currencies.

This experiment has inspired a foundational part of this project. This project has implemented a government-backed UBI scheme in which holders of the token are incentivised to spend (have a similar negative carry to that in the Wörgl experiment) so that the token has the maximum impact it can on the real economy. It does this by topping up a user's balance to a pre-set amount. Holders of the UBI token lose out if they don't spend their entire holdings and thus they are incentivised to spend.

Chapter 3

Smart Contract Design

This chapter explores the implementation of WörglCoin using the Ethereum blockchain. It will cover the logic of the smart contracts that have been developed, the security measures that have been introduced and the methodology that has been implemented for cost optimisation.

3.1 An Overview of Ethereum

In 2009, Bitcoin introduced a new type of digital currency, underpinned by a distributed ledger - the blockchain. In 2013, Ethereum expanded on this concept by “building a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications” [22]. In essence, smart contracts enable users to store and transfer value based on pre-set conditions written in code.

One of the core elements of Ethereum is the Ethereum Virtual Machine (‘EVM’). The EVM is essentially a run-time environment for smart contracts on the Ethereum network. Developers can use a high-level language (e.g. Solidity) to develop programs which are compiled into EVM byte-code and executed across every node in the network.

Every single EVM instruction must be paid for in terms of a specific number of units of ‘gas’. Each unit of gas has a price which is expressed in terms of Gwei (1 Ether = 1,000,000,000 Gwei). This execution cost is decided by the miners, which can refuse to process transactions under a certain gas price, and is designed to prevent Denial-of-Service attacks on the Ethereum network.

The EVM is a stack-based virtual machine with a memory byte-array and key-value storage. All elements on the memory stack are 32-bytes, and all keys and values in storage are 32 bytes.

3.2 Smart Contract Programming

All of the Ethereum smart contract code in this project has been written in Solidity [23]. This is the language of choice for most developers of decentralised applications. There are a number of other choices out there including Vyper [24] (which promotes security within smart contracts), LLL [25] (a low-level language similar to Assembly) and Flint [26].

Solidity has a number of similarities with traditional programming languages such as C, C++ or Javascript. A major difference to the Python programming language is that variables must be type cast.

As it stands today there are a number of limitations within Solidity that a developer has to deal with; for example, there is no support for floating point numbers. However, Solidity is evolving very quickly with each new iteration of the language released.

In general there are three key guiding principles of programming a smart contract. Firstly, the contract should be simple so that it can be reviewed and understood easily by the community. Secondly, the contract should display best-in-class security practices. Thirdly, the contract should be optimised to consume the least amount of gas as possible by minimising the number of individual operations being executed by the EVM.

3.3 Functionality

The smart contract of WörglCoin can be split into four main elements: contract attributes, modifiers, events, and functions.

Contract Attributes

In a similar manner to typical object-oriented programming languages, contract attributes can be seen as the state of the contract. These attributes can be of simple type (e.g. an unsigned integer), an array, a structure (which is a self-created object type) or a mapping (similar to a hash table with a key and corresponding value).

The system underpinning WörglCoin can be ultimately seen as an e-commerce site with an in-built currency (the UBI coin) distributed to registered users. The smart contract state includes information on consumers, businesses, orders and items.

For consumers, there is information on whether the account has been activated (whether the Ethereum address has performed identification measures to be an approved consumer), the UBI token balance (just a simple unsigned integer) and all orders the consumer has made (an array of unsigned integers corresponding to order IDs).

```
struct Consumer {
    bool isSet;
    address consumerAddress;
    uint tokenBalance;
    uint[] allOrders;
}
```

Figure 3.1: The consumer structure used in the WörglCoin smart contract.

For businesses there is information on whether the business address has been activated, the UBI token balance, all items supplied by the business (an array of unsigned integers corresponding to item IDs), whether a complaint has been lodged against the company, the number of outstanding complaints, all orders made to the company and the name of the company.

```
struct Business {
    bool isSet;
    address businessAddress;
    uint tokenBalance;
    uint[] itemsSupplied;
    bool complaintAgainst;
    uint noOfComplaints;
    uint[] allOrders;
    string name;
}
```

Figure 3.2: The business structure used in the WörglCoin smart contract

Within the contract there are also a number of attributes that make it easier to access the details of consumers, businesses, items and orders. This is typically achieved by having a mapping of an address / unsigned integer to the structure and then an array of addresses / unsigned integers that can be iterated through.

```
mapping(address => Consumer) public consumerDetails;
mapping(address => Business) public businessDetails;
mapping(uint => Item) public allItems;
mapping(uint => Order) public allOrders;

address[] consumerAddresses;
address[] businessAddresses;
uint[] itemIDs;
uint[] orderIDs;
```

Figure 3.3: The typical structure used to keep track of key elements

It is also key to remember that by using a public, permissionless Blockchain such as Ethereum, all attributes are accessible to the outside world (even if the ‘private’ keyword is used). Thus, the idea of privacy within an Ethereum smart contract is redundant and therefore this is a key consideration when it comes to identification and personal details.

Modifiers

In Solidity, modifiers can be used to alter function behaviour and enforce their correct use. For example, modifiers can be used to ensure that a function will only be executed by the contract owner, such as withdrawing funds or making changes to key state variables.

Within the smart contract there are modifiers to check whether the function is being executed by the contract owner (e.g. the government backing the coin), whether a

registered consumer is executing a function or whether a business is executing the function. This ensures that only the correct registered user can perform key actions such as ordering items and receiving Ether in exchange for the UBI coin.

Furthermore, functions can have three further modifiers pre-defined within Solidity - pure, view and payable. A function marked with the keyword 'pure' means that the function is not allowed to read or write to internal storage (modify the contract attributes). A function marked with the keyword 'view' means that the function is allowed to read, but not write to, internal storage.

Importantly, a function which is either 'pure' or 'view' can be called through web3 (the Javascript library for interacting with smart contracts) without needing a transaction. Therefore, these function calls are completely free and instantaneous.

A function marked with the keyword 'payable' allows Ether to be passed into the contract as part of the function call. Without this modifier, the transaction would be rejected.

```
modifier isOwner() { require(msg.sender == master); _; }  
  
modifier isConsumer() { require(consumerDetails[msg.sender].isSet); _; }  
  
modifier isBusiness() { require(businessDetails[msg.sender].isSet); _; }
```

Figure 3.4: Modifiers ensure functions are being executed as intended

Events

In Ethereum, events allow smart contracts to make interactions with the outside world. When a particular transaction is mined on the Ethereum network, smart contracts can emit events (through the transaction logs) which the front-end application can then process and respond to.

In the WörglCoin application, the front-end needs to update when a new order is made, a new consumer or business registers and a variety of other events occur. The front-end will listen for particular events which are triggered within certain functions of the smart contract.

```
event ConsumerAdded(address consumerAddress);
event BusinessAdded(address businessAddress);
event ItemAdded(uint itemID);
event OrderAdded(uint orderID, address businessAddress);

event TokenDistribution();
event TopUp();

event ConsumerChange(address consumerAddress);
event BusinessChange(address businessAddress);
event ItemChange(uint itemID);
event OrderChange(uint orderID, address businessAddress);
```

Figure 3.5: Events trigger updates to the user interface

Functions

The functions with the smart contract execute the main logic of the UBI system. In Solidity there are four distinct types of function - public, external, internal and private.

Public functions can be executed by anybody, whereas external functions can only be accessed externally and not by the contract itself. Internal functions (marked with the keyword 'internal') can only be called inside the current contract and all contracts derived from it. Private functions can only be accessed from within the contract.

3.4 Security

Smart contract security is a key area of research today given the large number of smart contract ‘hacks’ that have taken place. The most famous of these took place in 2016 when a hacker was able to exploit the DAO smart contract to drain c.3.6m Ether (worth approximately US\$70m at the time) in a few hours [27]. This section will cover the main security exploits [28] and how WörglCoin has been designed to mitigate against these risks.

Re-Entrancy

In order to understand this vulnerability it is first important to understand what is meant by a ‘fallback function’. In Solidity, a contract may have exactly one unnamed function which has no arguments and no return values. This function is executed in one of three circumstances: 1) if no function in the contract matches the name specified in the contract call, 2) if no data is supplied, or 3) if the contract receives plain Ether without any corresponding data.

This particular attack occurs when a smart contract is sending Ether to an unknown address. By carefully constructing a smart contract, the malicious attacker can invoke code in their fallback function when it receives funds from your smart contract. This malicious code then executes functions of your smart contract before the completion of the original code execution.

In the WörglCoin smart contract, Ether is sent to the address of a business when they have received a number of UBI tokens. In order to prevent a re-entrancy attack, the in-built ‘transfer’ function is used which only sends 2,300 gas. This is not enough gas for the destination address to call another contract.

The second precautionary measure that has been taken is to ensure that all state variable changes occur before transferring Ether to the destination address. This ensures that any re-entrancy attack could not take advantage of partial execution of function logic.

Arithmetic Over/Under Flows

As discussed earlier in this chapter, the EVM specifies fixed-size data types for integers. Classic computer programming arithmetic over- and under-flows can lead to many problems in function execution if user input is not sanitised correctly or the final state of the output is not checked thoroughly.

In order to prevent such an attack, the ‘SafeMath’ library [29], created by OpenZeppelin, has been used. This library replaces the traditional arithmetic operations with ‘safe’ math operations which validate the output is as expected.

```
function resetTokenBalance() public isOwner payable {

    // Reset all consumer balances
    for(uint i = 0; i<noOfConsumers; i++) {
        consumerDetails[consumerAddresses[i]].tokenBalance = topUpLevel;
    }

    // Pay out all funds to business
    for (uint j = 0; j<noOfBusinesses; j++) {
        businessDetails[businessAddresses[j]].tokenBalance = 0;
        businessAddresses[j].transfer
            (mul(businessDetails[businessAddresses[j]].tokenBalance,
                tokenValue));
    }

    emit TokenDistribution();
}
```

Figure 3.6: Measures taken to prevent a ‘re-entrancy’ attack

```
function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
    if (a == 0) {
        return 0;
    }

    c = a * b;
    assert(c / a == b);
    return c;
}
```

Figure 3.7: OpenZeppelin’s safe arithmetic operations library

Unexpected Ether

This vulnerability occurs when a developer assumes that Ether sent to a contract either executes the fallback function or another payable function. A developer may then have requirements in function that check ‘this.balance’ to verify whether certain logic can execute. However, if Ether has been forcibly sent to a smart contract, a malicious attacker can change the intended logic execution.

Given that the WörglCoin smart contract allows the owner to send Ether to the contract, the smart contract implements a state variable ‘balance’ which is changed only when the intended payable function is executed. This variable will never be manipulated by Ether forcibly sent to the contract and therefore all checks utilising this variable can be deemed reliable.

Delegatecall

In order to understand this vulnerability two opcodes implemented in the EVM need to be understood: CALL (0xf1) and DELEGATECALL (0xf4). External call messages are handled by the EVM's CALL operation where the context of the call switches to the external contract. The DELEGATECALL opcode is identical in execution except that the context remains with the calling contract (i.e. the msg.sender is kept as the caller).

Given that this vulnerability typically arises in the context of 'libraries' within Ethereum, the Wörglcoin smart contract does not require any mitigations against this vulnerability.

Default Visibilities

As discussed earlier, functions can have several different types - public, external, internal and private. If no type is specified then Solidity defaults to 'public'. This particular vulnerability occurs when developers allow functions to default to public that should in fact only be used in the current contract.

Recent versions of the Solidity Compiler will actually give warnings about functions without an explicit type defaulting to public. Furthermore, it is always safe practice to explicitly type the functions outlined in your contract.

Denial Of Service (DOS)

In a similar manner to Denial of Service attacks on web applications, smart contracts can be vulnerable to such an attack. A malicious user may be able to leave a contract inoperable for a small period of time or permanently. Typically this vulnerability can occur if a contract interacts with an externally controlled state variable or an outside data source (such as an Oracle).

There are many different ways to prevent these kind of attacks (given they are very broad). Given that this smart contract loops through arrays of consumers and businesses signed up to the site, in theory a malicious user could register a vast number of users and make this looping very expensive.

Entropy Illusion

There is no randomness within Ethereum and thus using any type of random number generation is a common source of problems. Many developers in the past have used block variables (for example the timestamp of a block) to introduce some concept of randomness into their system. However, these variables are, in essence, controlled by the miner and are therefore not random.

A common solution to this problem is to use an oracle service (a data provider intermediary that brings off-chain data on-chain) to provide random numbers in your application. Given that the smart contract does not require random number generation, this vulnerability is not relevant.

Race Conditions / Front Running

This vulnerability takes advantage of the fact that in the Ethereum network transactions are pooled together inside ‘blocks’. Miners choose which transactions (those with the highest gas limit) get included in the mined block. A malicious attacker can watch these transactions, see which ones are not mined and attempt to front-run these transactions by executing their own with a higher gas price.

There are no front-running vulnerabilities in this smart contract given there is no ‘hidden information’ revealed that users can take advantage of. A number of preventative measures exist but there is no definitive way to protect against a malicious miner.

External Contract Referencing

Developers are able to explicitly reference and execute functions on contracts already deployed on the Ethereum mainnet. A problem can occur if the contract address being executed contains malicious code or has unintended logic execution.

The simplest way to prevent this is for the developer to hardcode the contract address in their smart contract and independently verify the code to be executed. In this smart contract no external contract calls are made and therefore no preventative measures have been taken.

Unchecked CALL Return Values

This vulnerability occurs when a developer does not check the boolean result of their ‘call()’ or ‘send()’ functions and assumes in their function logic that the transaction has succeeded.

In order to prevent such a vulnerability, this smart contract utilises the ‘transfer()’ method which will revert if the external transaction reverts. If ‘call()’ or ‘send()’ methods are being used than a simple ‘require’ statement can be used to check the boolean return value.

Uninitialised Storage Pointers

This vulnerability occurs when uninitialised local storage variables accidentally point to other storage variables in the contract. In essence this is similar to an overflow problem experienced in traditional computing.

In order to prevent such a vulnerability the developer should explicitly declare whether the variable is memory or storage when dealing with complex types.

Block Timestamp Manipulation

Typically to get access to the date or time in a smart contract a developer will use the timestamp of the block that the transaction has been mined in (using the function 'now' in the smart contract). However, the vulnerability exists as miners are able to change the timestamp of a block and gain advantageous execution of the smart contract logic in some way.

In order to prevent such an attack it is recommended to use either the block number (which will always be monotonically increasing) or take an average of block timestamps. In this smart contract, no concept of time is used as the smart contract owner distributes new funds when they would like, not based on any pre-determined timing condition.

3.5 Optimisation

When building a system based on the Ethereum network there are two main costs to consider. The first is the cost of initially deploying the contract on the network. The second is the ongoing gas cost associated with executing the functionality of your contract.

Initial Deployment

Initialising and deploying your smart contract is the most expensive part of the system. Part of the reason for this is the fact that the opcode ‘SSTORE’ is used frequently to initialise the storage for contract attributes. For the Wörglcoin smart contract, the initial creation and deployment costs approximately US\$183.

In terms of optimising the initial deployment of the smart contract, the Solidity compiler makes a number of initial optimisations. For example, all variable names and comments are removed from the code prior to deployment.

Storage is the most expensive component of Ethereum smart contracts. One way to optimise storage within a smart contract is to employ a technique called ‘packing’. If certain variable types are put together in a structure, the values will be stored consecutively in the same 32-byte storage word.

Function Execution

Given that the ongoing cost to the users of the Wörglcoin system will be function execution (particular focus is placed on consumers who will regularly be buying items), each and every function needs to be optimised to consume as little gas as possible.

The first optimisation technique that can be used is the ordering of logical comparisons. As with most other programming languages, the first statement will be executed before the second. Therefore, the more expensive computation should be placed second in the comparison as there is a chance it may not be executed.

The second optimisation technique that can be employed is to minimise input and output operations (less reads and writes to the storage variables). This is particularly important when it comes to expensive loop operations in a smart contract. By declaring a local variable that is the focus of a loop (rather than the storage variable), there will be only one read and one write operation.

The third optimisation technique is to use as small data types as possible. For example, if you know your ‘bytes’ type will be a certain length, it will be significantly cheaper to use ‘bytes1’ or ‘bytes32’ compared to ‘bytes’ in a smart contract.

The final optimisation technique that can be used is the correct use of the ‘memory’ storage variable. Memory variables are temporary variables and exist only in the scope of the function call. Given that they require no long-term storage, memory variables are fairly cheap to use. This is particularly useful in this smart contract when new consumers, businesses, orders and items are created.

Gas Price (Gwei)	3
Gas Price (Ether)	0.000000003
USD Per Ether	228.12
Gas Price (USD)	0.00000068436

Function	Gas Units	Cost Bearer	Cost (Gwei)	Cost (Ether)	Cost (USD)
Contract Creation + Deployment	266,666,667	Government	800,000,000	0.80	182.50
addBusiness	159,465	Government	478,395	0.00	0.11
addConsumerHash	44,755	Government	134,265	0.00	0.03
buyItem	189,970	Consumer	569,910	0.00	0.13
changeOwner	28,882	Government	86,646	0.00	0.02
changeTokenValue	28,176	Government	84,528	0.00	0.02
consumerSignUp	3,142,076	Consumer	9,426,228	0.01	2.15
makeComplaint	67,351	Consumer	202,053	0.00	0.05
markOrderAsSent	66,824	Business	200,472	0.00	0.05
changeTokenBalance	28,270	Government	84,810	0.00	0.02
resetComplaint	28,146	Consumer	84,438	0.00	0.02
resetTokenBalance	31,139	Government	93,417	0.00	0.02
sellItem	291,114	Business	873,342	0.00	0.20
topUpContract	43,181	Government	129,543	0.00	0.03

Figure 3.8: Gas costs of the Wörglcoin smart contract

3.6 Testing

In order to ensure the system of smart contracts is robust and performs as expected, an aggregated unit test has been created using a Javascript-based testing framework within Truffle (based on Mocha). Within this framework, a mixture of black-box testing (simulating interaction with the front-end application) and white-box testing (to thoroughly test the inner-workings of the contract) has been used. The results of these unit tests can be found in Figure 3.9.

In terms of code coverage, a tool entitled ‘Solidity coverage’ [30] has been used. This tool reports generic code coverage metrics including statement coverage, branch coverage, function coverage and line coverage. As seen in Figure 3.10, code coverage is close to 100% for statements, over 90% for branches and close to 100% for functions.

This testing framework demonstrates that there are no low-level generic errors (generally caught by initial deployment of the smart contract). Furthermore, given that several of the unit tests concentrate on the interaction between the smart contract and the front-end, there are no high-level, functional errors, with the smart contract meeting all the needs of both consumers and businesses.

```

dyn3145-144:Worgl-Coin MatthewMorrison$ truffle test
Using network 'development'.

Compiling ./contracts/Verifier.sol...
Compiling ./contracts/WorglCoin.sol...

Contract: WorglCoin
  ✓ The contract owner should be correctly identified (68ms)
  ✓ The starting number of consumers should be 0
  ✓ The starting number of businesses should be 0
  ✓ There should be no orders for an address that is not signed up
  ✓ An address that is not signed up should return 'Not Signed Up'
  ✓ Nobody apart from the contract owner should be able to change ownership (44ms)
  ✓ The owner of the contract should be able to change ownership (138ms)
  ✓ Only the contract owner should be able to add a Business (53ms)
  ✓ Only the contract owner should be able to add an eligible consumer
  ✓ A consumer that enters the wrong details should not be able to sign up (140ms)
  ✓ A consumer that does not have the right zero knowledge proof cannot sign up (pairing mistake) (803ms)
  ✓ A consumer that does not have the right zero knowledge proof cannot sign up (input mistake) (178ms)
  ✓ A consumer that enters the right details should be able to sign up (13325ms)
  ✓ A consumer should not be able to sign up twice. (73ms)
  ✓ The contract owner should not be able to add a duplicate consumer hash. (52ms)
  ✓ The contract owner should be able to successfully add a Business (323ms)
  ✓ A business should not be allowed to sign up twice. (50ms)
  ✓ The contract owner should be able to change the top up balance and value (258ms)
  ✓ Only the contract owner should be able to reset everyone's token balance (134ms)
  ✓ No other account should be able to add an item for sale. (91ms)
  ✓ A business should be able to add an item for sale if they have no complaint against them. (356ms)
  ✓ Only a registered consumer should be able to buy an item. (53ms)
  ✓ A consumer should be able to buy an item if they have enough tokens. (259ms)
  ✓ A consumer should not be able to buy an item if they do not have enough tokens. (207ms)
  ✓ A random business should not be able to mark an order that isn't theirs as sent. (50ms)
  ✓ A consumer should not be able to make a complaint about an order if it hasn't been marked as sent. (52ms)
  ✓ A consumer should not be able to make a complaint about an order if it isn't their order. (57ms)
  ✓ A business should be able to mark an order as sent. (230ms)
  ✓ A consumer should be able to make a complaint about an order if it has been mark as sent. (125ms)
  ✓ A business with a complaint against it should not be able to put another item up for sale. (52ms)
  ✓ A random consumer should not be able to withdraw their complaint about another order. (57ms)
  ✓ The right consumer should be able to withdraw their complaint about a business. (135ms)
  ✓ The owner of the contract should be able to fund the contract. (77ms)
  ✓ A business should be transferred the right amount of Ether after the owner resets the contract. (832ms)

34 passing (19s)

```

Figure 3.9: Smart contract unit testing

all files / contracts/ WorglCoin.sol**99.2%** Statements 124/125 **92%** Branches 46/50 **100%** Functions 30/30 **99.3%** Lines 141/142

```
1  pragma solidity ^0.4.23;
2
3  import "./Verifier.sol";
4
5  contract WorglCoin {
6
7      /*****
8      /***** CONTRACT ATTRIBUTES *****/
9      /*****/
10
11     struct Consumer {
12         bool isSet;
13         address consumerAddress;
14         uint tokenBalance;
15         uint[] allOrders;
16     }
17
18     struct Business {
19         bool isSet;
20         address businessAddress;
21         uint tokenBalance;
22         uint[] itemsSupplied;
23         bool complaintAgainst;
24         uint noOfComplaints;
25         uint[] allOrders;
26         string name;
27     }
28
29     struct Order {
30         uint orderID;
31         uint itemID;
32         address customerAddress;
33         uint quantityOrdered;
34         bool sent;
35         uint delivery_location;
```

Figure 3.10: Smart contract code coverage

Chapter 4

Identification

This chapter explores one of the major difficulties in implementing a cryptocurrency UBI system: unique identification of citizens. An overview of solutions in existing cryptocurrency UBI schemes and their flaws is presented. Furthermore, the use of zero-knowledge proofs in the Wörglcoin UBI system is explained, as well as the mathematics behind their implementation.

4.1 The Identification Problem

A UBI system cannot be susceptible to Sybil attacks. A Sybil attack is where malicious users create fraudulent accounts in order to subvert and impede the functionality of the system. The UBI system implemented in this project aims to distribute tokens to unique individuals, therefore a robust system of identification is required.

All government entities worldwide have a significant volume of pre-existing data on citizens. For example, in the UK the government has our National Insurance number, our permanent residential address, age, and much more. Typically when a citizen is signing up for a new government service they will be required to enter their information and this will be verified against the government's records. For added security, the government typically sends a unique code to the individual's permanent residential address.

The UBI system presented in this project allows a user to sign up using these traditional data points that the government holds. However, there is one fundamental problem when using a public, permissionless blockchain such as Ethereum: all transactions and corresponding function parameters are public. Therefore, any malicious party monitoring the network could be privy to all personal data disclosed during the sign up process.

4.2 Existing Solutions

There are a number of solutions that have been utilised by existing cryptocurrency UBI projects. These existing solutions have several advantages and disadvantages which will be explored below.

Mobile Phone Identification

Several cryptocurrency UBI schemes, including Mannabase, use SMS-based authentication to ensure that fraud is not committed on their site. When a consumer signs up for an account on Mannabase, they are required to enter their mobile phone number, after which they will receive an SMS message with a code for verification purposes.

However, the security of SMS-based verification has been questioned in recent years. In July 2017 the National Institute for Standards and Technology in the US advised abandoning SMS-based two-factor authentication given the risk of interception [31]. This is one of the reasons that Google now uses an application for two-factor authentication rather than SMS.

As well as being a security risk, many individuals worldwide have multiple mobile phones (typically for leisure and work purposes). This means that Mannabase could potentially face a single individual with multiple mobile devices signing up for multiple accounts, thus undermining the distribution of their token.

Passport Verification

Since 2008, more than 60 countries have issued biometric passports containing a RFID chip, which can be easily read by certain devices, including smartphones. The 'UBIC' project, for example, uses the unique digital signature of an E-Passport to verify the identity of an individual. This unique digital signature is then cryptographically linked to a UBIC wallet address.

There have been many public displays of security vulnerabilities in E-Passport architecture, including the ability to copy all the data to a smartcard using a standard contactless card interface and a simple file transfer tool [32]. Furthermore, it is fairly trivial to use a stolen passport to scan and subscribe to these services.

Social Profile Vetting

Another method utilised for individual verification is to ask users to submit a social profile (e.g. Facebook) at the time of sign-up. This is a technique employed by the 'BIG' foundation [33] where a user will submit their social profile and it will be vetted by an individual at the foundation.

As well as not being particularly scalable given the human verification required, it is common knowledge that social media companies host many fake and duplicate accounts in their systems. At the end of 2017, Facebook admitted in its earnings call that around six to ten percent of accounts on the site are duplicate and two to three percent are fake [34]. This amounts to as many as 270 million accounts on the site that are duplicate or fraudulent. This is clearly a problem for systems that rely on verification through social media profiles.

Biometrics

Several UBI schemes have proposed using biometrics (e.g. DNA) to uniquely identify individuals. However, these solutions are intrusive and open to misuse (e.g. getting hold of someone else's DNA and using it for sign-up). Furthermore, given that the technology is at a very early stage, most UBI schemes have noted that these systems will not be possible in the near-future.

User Recommendations

Another method in place today at 'SwiftDemand' is the use of 'identity providers' who are responsible for validating the identity of new individuals and signing them up to the service. There are clear limitations and vulnerabilities of such a system, including how vulnerable the system is to a handful of rogue actors undermining the identification process.

4.3 Interactive Zero-Knowledge Proofs

An interactive zero-knowledge proof is a cryptographic construct that allows one party ('Peggy') to prove to another party ('Victor') that they know a public value 'x', without revealing any private information that leads them to knowing 'x'. For example, 'Peggy' could prove to 'Victor' that they have all of the inputs required to create a public SHA256 hash value, without disclosing any of the inputs.

A zero-knowledge proof must satisfy three conditions: 1) completeness, 2) soundness, and 3) zero-knowledge. Completeness dictates that if a statement is true then the verifier will be convinced of this fact by an honest prover. Soundness means that an honest verifier cannot be tricked by a cheating prover. Zero-knowledge means that no information is leaked to the verifier as a result of verifying the statement to be true.

Zero-knowledge proofs are examples of 'interactive proof systems' in which provers and verifiers can exchange challenges and responses. Interactive proof systems are probabilistic in nature but the probabilities can be made arbitrarily close to 100%.

A very good practical demonstration of an interactive zero-knowledge proof is the 'Ali Baba cave' [35]. There is one entrance to this cave with two routes, *A* and *B*. The two routes of the cave are separated by a door that can only be unlocked with a secret key, *S*.

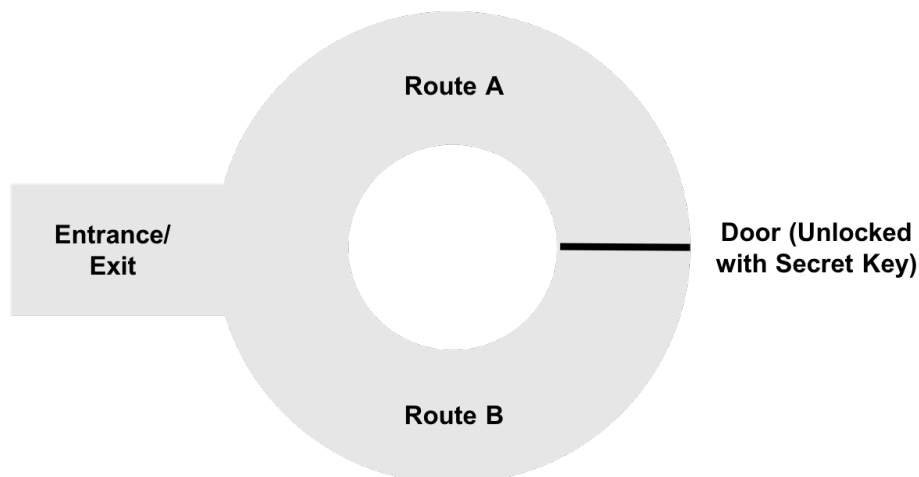


Figure 4.1: The Ali Baba cave

In this example, Peggy (the ‘prover’) has access to the secret key S and wants to prove this to Victor, the ‘verifier’. Peggy wants to prove to Victor that she knows the secret key, without actually revealing the secret key.

In order to do this, Peggy enters the cave and, with random probability and unknown to Victor, chooses either path A or path B . Victor then stands outside the cave and shouts the name of the path that he wants Peggy to come down.

If Peggy knows the secret key then this is an easy task. She simply opens the door and exits the path as chosen by Victor. However, if she does not have the secret key, then the probability is only 50% that Peggy will exit by the right path. By simply repeating this experiment many times, the probability shifts to 25%, 12.5% etc. until Victor can be almost certain Peggy knows the secret key.

4.4 Non-Interactive Zero-Knowledge Proofs

A non-interactive zero-knowledge proof does not require any interaction between the ‘prover’ and ‘verifier’. An article published in 2012 by Bitansky, Canetti, Chiesa and Tromer [36] introduced the acronym zk-SNARK for a zero-knowledge succinct non-interactive argument:

- **Succinct**: the messages have a relatively small size, especially compared to the complexity of the computation required.
- **Non-interactive**: there is no interaction between the prover and the verifier.
- **ARguments**: the verifier is only protected against computationally limited providers.
- **Knowledge**: it is not possible for the prover to construct a proof without knowing a witness.

Quadratic Arithmetic Program

The type of problem that a zk-SNARK can be applied to is called a ‘quadratic arithmetic program’ (‘QAP’) [37]. In the following example a prover will want to demonstrate that they know the solution to the quadratic equation $x^2 + 2x + 3 = 27$. The solution that our prover will have to know is that $x = 4$ or $x = -6$.

Stage 1: Define Problem in High-Level Language. The problem is first described in terms of a high-level language supporting basic arithmetic:

```
def quadratic(x):  
    /* x is a private variable, in this case the solution to the quadratic  
    equation */  
    y == x**2  
    return y + 2 * x + 3
```

Stage 2: Flattening. To begin the process of converting this high-level language code into a QAP, the code needs to be processed through a stage called ‘flattening’. This is where the code outlined above is converted into statements of the form $x = y$ or $x = y \text{ (operation) } z$, where *operation* can be an arithmetic operation of the kind $+$, $-$, $*$, or $/$. This is sometimes referred to as creating a ‘circuit’ as each line of code as a result of our flattening can be likened to a pass through one kind of logic gate.

Flattening our above high-level language problem:

```
y = x * x
sym_1 = 2 * x
sym_2 = y + sym_1
~out = sym_2 + 3
```

Stage 3: Convert Flattened Code to R1CS. After flattening this problem, the circuit is then converted into a rank-1 constraint system (‘R1CS’). An R1CS is a sequence of groups of three vectors (a , b and c), with a solution vector s that solves the following:

$$s \cdot a * s \cdot b - s \cdot c = 0 \quad (4.1)$$

An R1CS constraint (these three vectors a , b and c) is actually created for every ‘logic gate’ step (every line of the ‘flattened’ code). Furthermore, there are standard ways of converting a ‘logic gate’ step into this sequence of three vectors, depending on the arithmetic operation being applied.

The final R1CS for the problem will be have four sets of three vectors, with each vector having a height equal to six (the number of variables in the problem). A detailed R1CS conversion from the flattened code can be seen in Figure 4.2.

The witness is then simply the solution vector (i.e. vector ‘S’) that solves each constraint in the system. In this case, the witness would simply be $[1, 3, 18, 9, 6, 15]$.

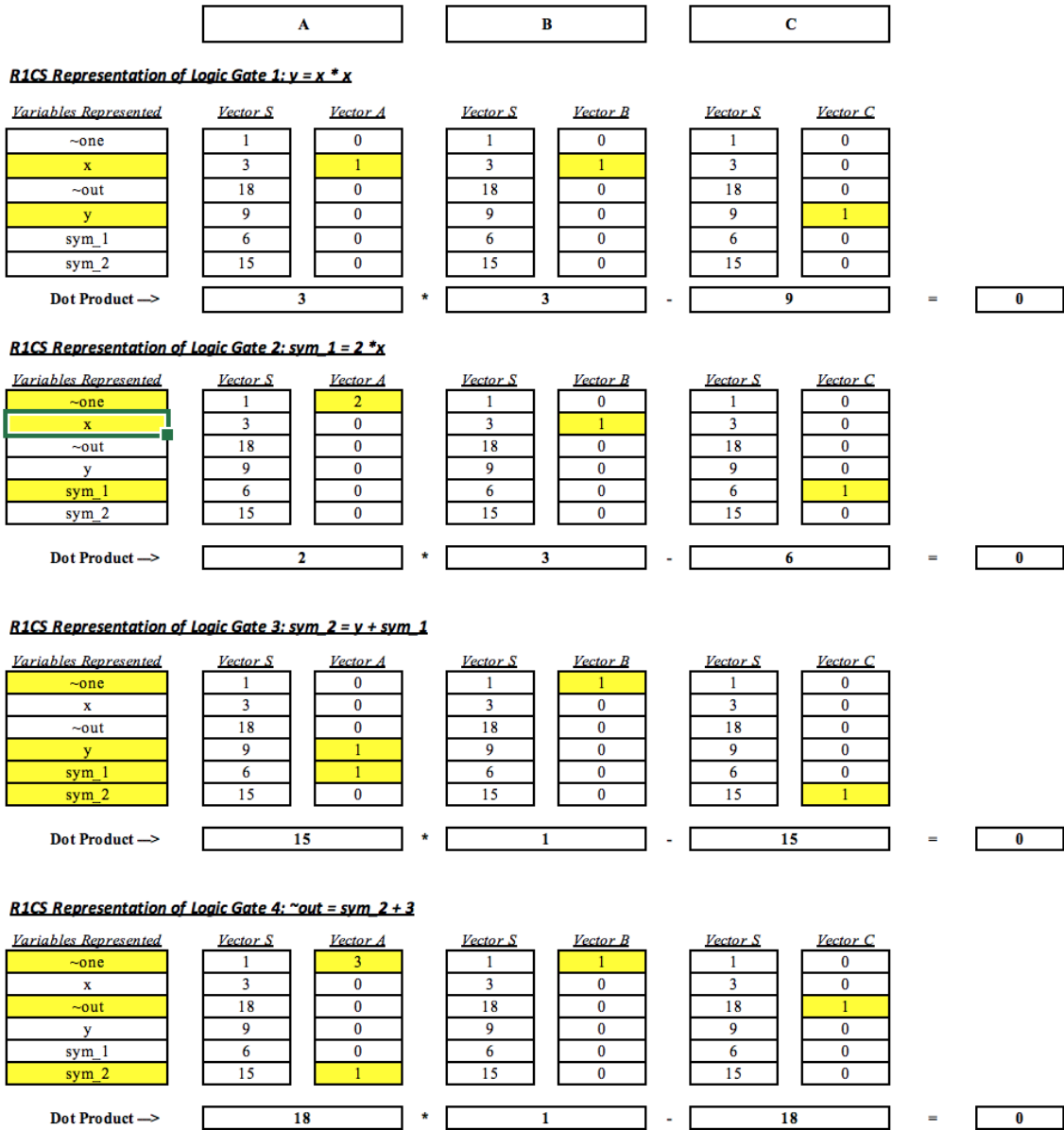


Figure 4.2: An R1CS representation of the flattened code

Stage 4: Convert R1CS into QAP Form. We then need to convert this R1CS into QAP form, by using polynomials instead of dot products. By using Lagrange interpolation, we go from n sets of three vectors, each with a height equal to the number of variables (h) to h sets of three degree-3 polynomials. Evaluating the polynomials at each x-coordinate then represents one of the n constraints in the system.

Lagrange interpolation is simply a method to find a polynomial that passes through all specified (x,y) coordinate pairs. In order to do this, the first value of every a vector is taken (in this example it would be $[0, 2, 0, 3]$), converted to coordinates where the x-coordinate represents the position and the y-coordinate represents the value (so in this case it would be $[(1, 0), (2, 2), (3, 0), (4, 3)]$) and then use Lagrange interpolation to find the polynomial going through these points.

This process is then repeated for the second, third, fourth, fifth and sixth elements of every a vector, and then repeated again for the b and c vectors. As a result, we end up with six sets of degree-3 polynomials for each vector. A detailed overview of the transformation to QAP can be seen in Figures 4.3 to 4.6.

The reason behind transforming the problem into QAP is that we can now check all of the constraints at the same time by doing the dot product check on the polynomials, as seen in Figure 4.7.

If the resulting polynomial is zero at every x-coordinate, that means all of the checks pass. In this system, if any fields in any of the R1CS constraints does not hold (i.e. a logic gate step does not hold), then the resulting polynomial will not evaluate to zero at every x-coordinate.

As can be seen in Figure 4.8, the resulting polynomial is equal to zero at every x-coordinate relevant to each logic gate step. Therefore, the prover has correctly generated each line of the problem and there this is a valid proof.

To speed up the process, instead of having to evaluate the new polynomial at every x-coordinate, we simply divide t by another polynomial (let's call it Z) and check that there is no remainder once you make this division.

The polynomial Z is defined as a polynomial that is equal to 0 at all relevant points for our logic gate (i.e. $x = 1$ to $x = 4$). Therefore, any polynomial that is equal to 0 at all of its points has to be some multiple of our polynomial Z . Therefore, if we find no remainder when making this division, we can be sure that our polynomial t is equal to 0 for all relevant x-coordinates.

Step 1: Take the values from each vector

Vector A, Position 1	0	2	0	3
Vector A, Position 2	1	0	0	0
Vector A, Position 3	0	0	0	0
Vector A, Position 4	0	0	1	0
Vector A, Position 5	0	0	1	0
Vector A, Position 6	0	0	0	1

Vector B, Position 1	0	0	1	1
Vector B, Position 2	1	1	0	0
Vector B, Position 3	0	0	0	0
Vector B, Position 4	0	0	0	0
Vector B, Position 5	0	0	0	0
Vector B, Position 6	0	0	0	0

Vector C, Position 1	0	0	0	0
Vector C, Position 2	0	0	0	0
Vector C, Position 3	0	0	0	1
Vector C, Position 4	1	0	0	0
Vector C, Position 5	0	1	0	0
Vector C, Position 6	0	0	1	0

Figure 4.3: Step 1 of the QAP transformation

Step 2: Convert these values to coordinates

Vector A, Position 1	(1,0)	(2,2)	(3,0)	(4,3)
Vector A, Position 2	(1,1)	(2,0)	(3,0)	(4,0)
Vector A, Position 3	(1,0)	(2,0)	(3,0)	(4,0)
Vector A, Position 4	(1,0)	(2,0)	(3,1)	(4,0)
Vector A, Position 5	(1,0)	(2,0)	(3,1)	(4,0)
Vector A, Position 6	(1,0)	(2,0)	(3,0)	(4,1)

Vector B, Position 1	(1,0)	(2,0)	(3,1)	(4,1)
Vector B, Position 2	(1,1)	(2,1)	(3,0)	(4,0)
Vector B, Position 3	(1,0)	(2,0)	(3,0)	(4,0)
Vector B, Position 4	(1,0)	(2,0)	(3,0)	(4,0)
Vector B, Position 5	(1,0)	(2,0)	(3,0)	(4,0)
Vector B, Position 6	(1,0)	(2,0)	(3,0)	(4,0)

Vector C, Position 1	(1,0)	(2,0)	(3,0)	(4,0)
Vector C, Position 2	(1,0)	(2,0)	(3,0)	(4,0)
Vector C, Position 3	(1,0)	(2,0)	(3,0)	(4,1)
Vector C, Position 4	(1,1)	(2,0)	(3,0)	(4,0)
Vector C, Position 5	(1,0)	(2,1)	(3,0)	(4,0)
Vector C, Position 6	(1,0)	(2,0)	(3,1)	(4,0)

Figure 4.4: Step 2 of the QAP transformation

Step 3: Find degree-3 polynomial solutions

Vector A	Constant	x Term	x2 Term	x3 Term
Position 1	-15.0000	24.5000	-11.0000	1.5000
Position 2	4.0000	-4.3333	1.5000	-0.1667
Position 3	0.0000	0.0000	0.0000	0.0000
Position 4	4.0000	-7.0000	3.5000	-0.5000
Position 5	4.0000	-7.0000	3.5000	-0.5000
Position 6	-1.0000	1.8333	-1.0000	0.1667

Vector B	Constant	x Term	x2 Term	x3 Term
Position 1	3.0000	-5.1667	2.5000	-0.3333
Position 2	-2.0000	5.1667	-2.5000	0.3333
Position 3	0.0000	0.0000	0.0000	0.0000
Position 4	0.0000	0.0000	0.0000	0.0000
Position 5	0.0000	0.0000	0.0000	0.0000
Position 6	0.0000	0.0000	0.0000	0.0000

Vector C	Constant	x Term	x2 Term	x3 Term
Position 1	0.0000	0.0000	0.0000	0.0000
Position 2	0.0000	0.0000	0.0000	0.0000
Position 3	-1.0000	1.8333	-1.0000	0.1667
Position 4	4.0000	-4.3333	1.5000	-0.1667
Position 5	-6.0000	9.5000	-4.0000	0.5000
Position 6	4.0000	-7.0000	3.5000	-0.5000

Figure 4.5: Step 3 of the QAP transformation

Step 4: Evaluate these polynomials to get original vectors

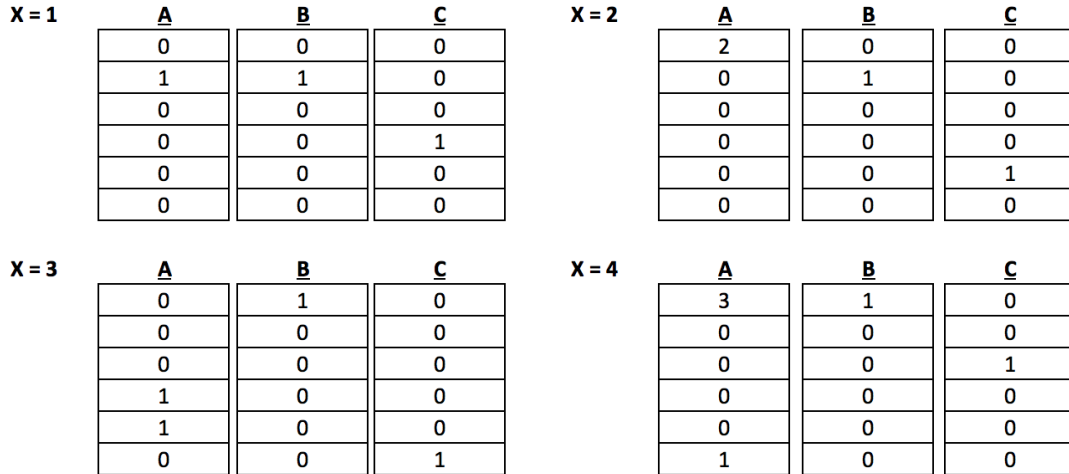


Figure 4.6: Step 4 of the QAP transformation

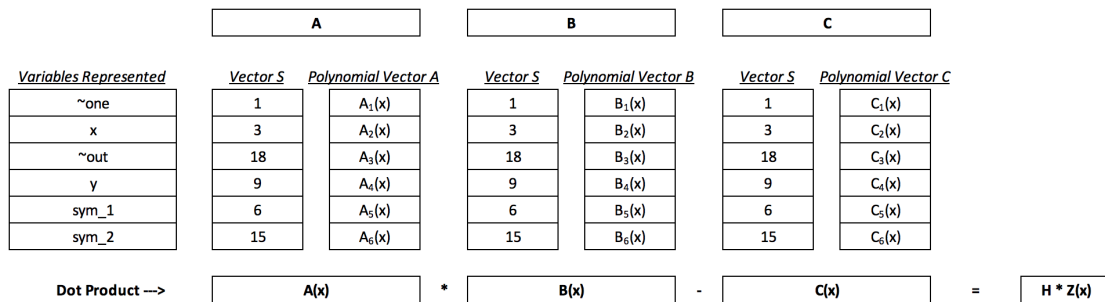


Figure 4.7: The QAP problem in terms of polynomials

A				
Vector S	Constant	x Term	x2 Term	x3 Term
1	-15.0000	24.5000	-11.0000	1.5000
3	4.0000	-4.3333	1.5000	-0.1667
18	0.0000	0.0000	0.0000	0.0000
9	4.0000	-7.0000	3.5000	-0.5000
6	4.0000	-7.0000	3.5000	-0.5000
15	-1.0000	1.8333	-1.0000	0.1667
Result				
	42.0000	-66.0000	31.0000	-4.0000

B				
Vector S	Constant	x Term	x2 Term	x3 Term
B1(x)	3.0000	-5.1667	2.5000	-0.3333
B2(x)	-2.0000	5.1667	-2.5000	0.3333
B3(x)	0.0000	0.0000	0.0000	0.0000
B4(x)	0.0000	0.0000	0.0000	0.0000
B5(x)	0.0000	0.0000	0.0000	0.0000
B6(x)	0.0000	0.0000	0.0000	0.0000
Result				
	-3.0000	10.3333	-5.0000	0.6667

C				
Vector S	Constant	x Term	x2 Term	x3 Term
0	0.0000	0.0000	0.0000	0.0000
0	0.0000	0.0000	0.0000	0.0000
0	-1.0000	1.8333	-1.0000	0.1667
0	4.0000	-4.3333	1.5000	-0.1667
0	-6.0000	9.5000	-4.0000	0.5000
0	4.0000	-7.0000	3.5000	-0.5000
Result				
	42.0000	-54.0000	24.0000	-3.0000

	Constant	x Term	x2 Term	x3 Term
A · s	42.0000	-66.0000	31.0000	-4.0000
B · s	-3.0000	10.3333	-5.0000	0.6667
C · s	42.0000	-54.0000	24.0000	-3.0000
t = A · s * B · s - C · s				
	-168.0000	686.0000	-1009.0001	693.3333
Z				
	24.0000	-50.0000	35.0000	-10.0000
h = t / Z				
	-7.0000	14.0000	-2.6667	

	Constant	x Term	x2 Term	x3 Term	x4 Term	x5 Term	x6 Term
t = A · s * B · s - C · s	-168.0000	686.0000	-1009.0001	693.3333	-240.3333	40.6666	-2.6667
x=1							
		0					
x=2							
		0					
x=3							
		0					
x=4							
		0					

Figure 4.8: The solution to the polynomial QAP problem

Elliptic Curve Pairings

An elliptic curve is simply a set of coordinates that satisfy the equation:

$$y^2 = x^3 + ax + b \quad (4.2)$$

This equation graphs to what can be seen in Figure 4.9. There are two key properties of this curve [38]. Firstly, the curve is perfectly symmetrical about the x-axis. This means that any point can be reflected about the x-axis and remain on the same curve. The second key property of an elliptic curve is that any non-vertical straight line drawn between two points will hit the curve at only three points.

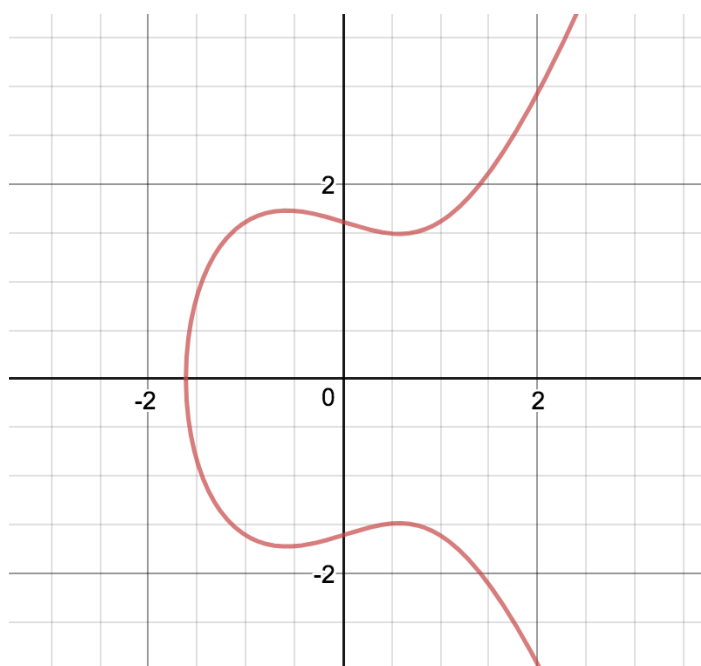


Figure 4.9: An elliptic curve

The second property is useful as a single point on the curve can be added to itself to create a new point. Adding a point to itself n times is a fairly trivial process. However, if you are given a final point on the curve and asked to calculate n , this is very difficult. This makes it suitable for cryptographic purposes.

When using elliptic curves, we tend to use modulo (typically with the maximum being a prime number) so that the numbers are restricted to a finite range. Therefore our elliptic ‘curve’ tends to look more like Figure 4.10.

The system of cryptography is then defined by picking a maximum number, a curve equation and a public point on this curve. The public key is the public point added to itself n times, where n is the private key.

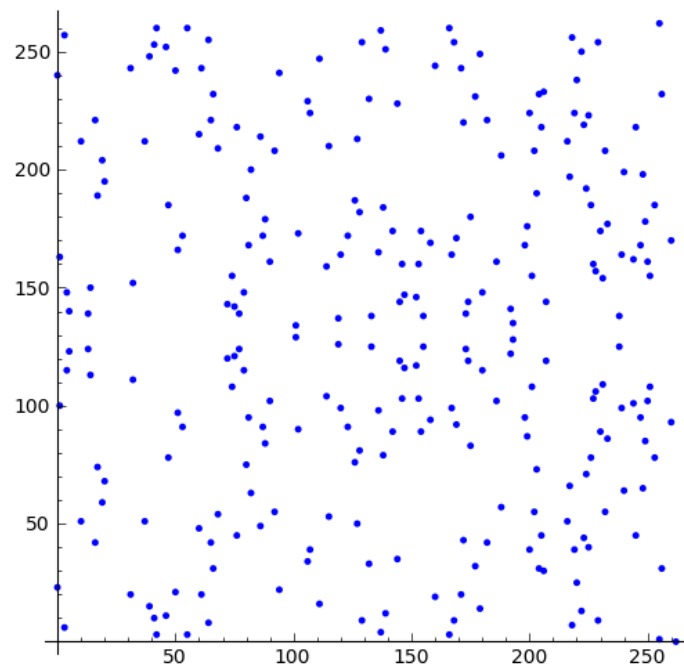


Figure 4.10: Point representation of an elliptic curve

Elliptic curve pairings allow you to check more complicated functions on points on the curve [39]. A pairing is denoted as $e(P, Q)$ (also known as a ‘bilinear map’) and has the following properties:

$$e(P, Q + R) = e(P, Q) * e(P, R) \quad (4.3)$$

$$e(P + S, Q) = e(P, Q) * e(S, Q) \quad (4.4)$$

It is possible to come up with a bilinear map where the inputs (i.e. P and Q) are elliptic curve points - this is called an ‘elliptic curve pairing’.

Basis of zk-SNARKs

Assume that we have an elliptic curve pairing, denoted (P, Q) , where $P * k = Q$ (nobody else knows k), and another elliptic curve pairing, denoted (R, S) where $R * k = S$ [40]. The ‘knowledge-of-exponent’ assumption, as outlined by Bellare and Palacio [41], means that the only way that (R, S) could have been created is by taking (P, Q) and multiplying both by some private factor r .

The solution to our QAP problem (the three polynomials we discussed earlier) are actually linear combinations of a set of polynomials. Using the ‘knowledge-of-exponent’ assumption, the prover can demonstrate that they have a linear combination of the polynomials, rather than the individual polynomials themselves.

As described by Vitalik Buterin [40], in order to do this we evaluate each of the polynomials at point t and multiply by some value k . Both of these variables should be completely private and discarded after the proof is generated.

The prover then needs to show that all three linear combinations have the same coefficients. In total, the verification process for a zk-SNARK requires an elliptic curve multiplication for each public input variable and five pairing checks.

4.5 Zk-SNARKs in Ethereum

At the time of writing, there are only two real projects aiming at creating a development framework for zk-SNARKs on Ethereum. The first, `libsnark` [42] provides a C++ implementation of creating an R1CS, an algorithm to generate proofs and an algorithm to check proofs for the statements. However, there is no direct method to convert the proof verification algorithm into a smart contract to be used on the Ethereum network.

Another project that has been developed for zk-SNARKs on Ethereum is `ZoKrates` [43]. `ZoKrates` provides a high-level language that compiles to an R1CS, a number of gadgets that have been exported from the `libsnark` library, as well as providing a method to export proof verifications in the form of a smart contract.

The `ZoKrates` library is run inside a Docker container and there are six main steps to create and execute zk-SNARKs using the `ZoKrates` library.

1. High-Level Language Circuit

The first step is to create the logic for verification. This corresponds to the function that will be executed by the prover to demonstrate they have the private variables required to generate the proof. In `ZoKrates` the keyword ‘private’ is used to denote private variables, with public variables requiring no keyword.

For example, if we were looking to create a circuit in which a prover would want to prove that they have a private variable x that, when added to three, generates a publicly known variable y , we would write the following code:

```
def main(private x, y):  
    x + 3 = y  
    return 1
```

2. Compilation / Code Flattening

The second stage is to compile the high-level language written in Step 1 to an arithmetic circuit (i.e. flattening the code). This is done through the `ZoKrates` command-line interface (‘CLI’) by typing `./zokrates compile -i /path/to/worglcoin.code`. This then creates a compiled file called `out.code` in the directory.

The steps to ‘flatten’ a complex hashing algorithm such as SHA256 is a fairly complex procedure. We cannot simply use SHA256 natively, we need to decompose it into several steps and operate on a bit-by-bit basis:


```

def main(bit1_public_hash,..., bit256_public_hash):
    /* check all the public inputs are bits */
    bit0_public_hash = bit0_public_hash * bit0_public_hash
    ...
    bit255_public_hash = bit255_public_hash * bit255_public_hash

    /* produce the sha256 hash */
    /* initialise hash values */
    H0 = 01101010000010011110011001100111
    ...
    H7 = 01011011111000001100110100011001

    /* check all the private inputs are bits */
    bit0_private_input = bit0_private_input * bit0_private_input
    ...
    bit511_private_input = bit511_private_input * bit511_private_input

    /* copy chunk into first 16 words w[0..15] of the message schedule
    array */
    w0b31,..., w15b0 = bit511_private_input, ..., bit0_private_input

    /* extend the first 16 words into the remaining 48 */
    for i from 16 to 63:
        s0 = (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor
            (w[i-15] rightshift 3)
        s1 = (w[i-2] rightrotate 17) xor (w[i-2] rightrotate 19) xor
            (w[i-2] rightshift 10)
        w[i] = w[i-16] + s0 + w[i-7] + s1

    /* initialise working variables */
    a = H0
    ...
    h = H7

    /* loop through compression function */
    for i from 0 to 63:
        S1 = (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate
            25)
        ch = (e and f) xor ((not e) and g)
        temp1 := h + S1 + ch + k[i] + w[i]
        S0 = (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate
            22)
        maj = (a and b) xor (a and c) xor (b and c)
        temp2 = S0 + maj

        h = g
        g = f

```

```
f = e
e = d + temp1
d = c
c = b
b = a
a = temp1 + temp2

/* add the compressed chunk to the current hash value */
h0 := h0 + a
...
h7 := h7 + h

/* assign bits to hash value */
bit0_private_hash,..., bit31_private_hash = h0
...
bit224_private_hash,..., bit255_private_hash = h7

/* check all bits are the same */
bit0_public_hash = bit0_private_hash
...
bit255_public_hash = bit255_private_hash

return 1
```

3. Computing the Witness

A witness is computed for the compiled program and all private and public variables are supplied by the ‘prover’. In order to do this the following is executed `./zokrates compute - witness - a 0 1 1 0 1 ...`. This creates a witness file which can be found at `./witness`.

4. Setting Up The Proof

This stage creates a trusted setup for the compiled program from step two. In order to do this the following is executed `./zokrates setup` and a proving key and a verifying key is generated.

5. Exporting the Verifier

ZoKrates then allows the exporting of the verification algorithm in the form of a smart contract. By executing `./zokrates export - verifier` a smart contract is created to verify the parameters generated in the preceding steps. This smart contract can be used for any witness computed on the compiled code.

6. Generating the Proof

The proof parameters are then generated by the ‘prover’ by executing `./zokrates generate –proof`. This will produce eight elliptic curve pairings which can be passed through to the smart contract for verification. An example of the parameters that are returned:

```
var A =
  ["0xbc58bed6b63e82175e9e3cf017f26d5f71a2f36165f3580d02514e3f2f22ae9",
   "0x30587068f5a1f61193afccf71f265e277de27e0651204b325b6625c2a51bd77e"];
var A_p =
  ["0x2437272ec7e61998347ed609e003d1bb9c069903c211776a6d0fc8db65906b82",
   "0x8df6216b168a62a4507df93468cf0e58b5221919ef86642541ddb54c15e589"];
var B =
  [ ["0x8d7ce8d510b51d0a25752103da900a7f3ab7ddfa2f73e2ba682a9139c25d1f7",
     "0x208deb2a232d751e5e0cac266ee43d55ff9c219a0b60cd1c61971cde841a705e"],
    ["0x623ac694f3992eaf7953c7827ceb6ba4da4075789b3679b74f9abe0aea444",
     "0xf505cf3d2fa6c24b218ef86254f83fb450f1b63f9c93373e865f04d8c5b936e"] ];
var B_p =
  ["0x19de1bdf2362573a606e72331c954183c9b98f8664d093682b73ab5f5be6d93a8",
   "0x35a943ce82ab2db74de5f6c68375de4029c0def09dae9245db7a18641b63125"];
var C =
  ["0x12bfe3da774a0dde4238ec240c136af4d0f089e5e769ba486255104a03c2e94b",
   "0x1e3418e825cb53d2eeef37fa587abf9bdfcc6415361f1f6a15731effcf9b2f58"];
var C_p =
  ["0x2c5f8013378ff540b33ae444d77dd17a407762f8940c05bb692ae13bdd5cec8f",
   "0x1a4fd227107bba27f57bbaa9e2338d990fd80dc6bb8a8dbc1bd69479cc737df3"];
var H =
  ["0x26c0d5e8b23bd7730636067406c59342081ca73cc6f2d2fb2acd6d676be80f1e",
   "0x419e311996e1c6d0aba5eba8d41357a64c92091873185a5e65bfad0186330f3"];
var K =
  ["0x539752f747c989c355aecf876d0f5aeae88796b9eed8709d4c212248be089ca",
   "0xa0a88a63c312f8dc82741c24885e6527a9f4e090d06c823151ebd7e9cea40cc"];
```

7. Executing the Smart Contract Proof

In order to verify these proof parameters, the above parameters and the public input parameters are passed through to the smart contract using a web3 call. For example, the following Javascript code would be executed from the front-end:

```

var A =
  ["0xbc48bed6b63e82175e9e3cf017f26d5f71a2f36165f3580d02514e3f2f22ae9",
   "0x30587068f5a1f61193afccf71f265e277de27e0651204b325b6625c2a51bd77e"];
var A_p =
  ["0x2437272ec7e61998347ed609e003d1bb9c069903c211776a6d0fc8db65906b82",
   "0x8df6216b168a62a4507df93468cf0e58b5221919ef86642541ddb54c15e589"];
var B =
  ["0x8d7ce8d510b51d0a25752103da900a7f3ab7ddfa2f73e2ba682a9139c25d1f7",
   "0x208deb2a232d751e5e0cac266ee43d55ff9c219a0b60cd1c61971cde841a705e",
   "0x623ac694f3992eaaf7953c7827ceb6ba4da4075789b3679b74f9abe0aea444",
   "0xf505cf3d2fa6c24b218ef86254f83fb450f1b63f9c93373e865f04d8c5b936e"];
var B_p =
  ["0x19de1bdf2362573a606e72331c954183c9b98f8664d093682b73ab5f5be6d93a8",
   "0x35a943ce82ab2db74de5f6c68375de4029c0def09dae9245db7a18641b63125"];
var C =
  ["0x12bfe3da774a0dde4238ec240c136af4d0f089e5e769ba486255104a03c2e94b",
   "0x1e3418e825cb53d2eeef37fa587abf9bdfcc6415361f1f6a15731effcf9b2f58"];
var C_p =
  ["0x2c5f8013378ff540b33ae444d77dd17a407762f8940c05bb692ae13bdd5cec8f",
   "0x1a4fd227107bba27f57bbaa9e2338d990fd80dc6bb8a8dbc1bd69479cc737df3"];
var H =
  ["0x26c0d5e8b23bd7730636067406c59342081ca73cc6f2d2fb2acd6d676be80f1e",
   "0x419e311996e1c6d0aba5eba8d41357a64c92091873185a5e65bfad0186330f3"];
var K =
  ["0x539752f747c989c355aecf876d0f5aeae88796b9eed8709d4c212248be089ca",
   "0xa0a88a63c312f8dc82741c24885e6527a9f4e090d06c823151ebd7e9cea40cc"];
var I = [0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
         1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1];

return WorglCoin.deployed().then(function(instance) {
  app = instance;
  return app.consumerSignUp(enteredHash, A, A_p, B, B_p, C, C_p, H, K,
    I, {from: consumer1, value: 0});
});
});

```

4.6 WörglCoin's Identification System

The UBI system presented in this project requires an individual to prove that they have the hash value of their private data (which the government can also construct), without revealing any of the corresponding information that has led to that hash in a public transaction. There are several main steps in WörglCoin's identification system, an outline of which can be found in Figure 4.11.

1. Contract Owner Supplies Hash

In the WörglCoin system, the contract owner is responsible for submitting eligible hashes. This will be the SHA256 compression function hash of an eligible citizen's name, national insurance, date of birth and unique code sent to their home address. Given the new nature of ZoKrates, all strings have to be converted to bits, appended to one another, padded to 512 bits and then the SHA256 compression algorithm is applied.

As a result, the smart contract will hold an eligible consumer's hash which is publicly visible but gives away no private details about a citizen.

2. A Consumer Enters Details

A new consumer which has not signed up to the site is then able to enter their details into the web application. In the background, without the user realising, the details are converted to bits, appended to one another, and then padded to 512 bits. This will act as the private input to the zero-knowledge proof.

3. Information Is Sent to Server

The consumer's 512-bit private information, as well as the first 32 bits of the public hash, is then sent to a server which is running a Docker container of the modified ZoKrates software.

The ZoKrates software has been amended somewhat so that it can verify that the first 32 bits of the public hash are equal to the first 32 bits of the SHA256 compression hash of the private information supplied in the stage of computing the witness.

The reason this has been done is that verifying all 256 bits of the hash would create too many constraints and thus make the on-chain, smart contract verification far too expensive. A four-byte check was used in line with many common applications, including the commonly used checksum (used within systems such as Bitcoin). However, as discussed in later sections, further work can be done to check all bits of the hash and therefore ensure the system is totally secure.

The node server then uses this information to compute a witness (Step 3 above)

and generate a proof (Step 6 above). The server then returns the proof variables back to the user's front-end.

4. An Ethereum Transaction is Triggered

Behind the scenes an Ethereum transaction is triggered in which the user's Ethereum address is used to call the 'verifyTx' function which verifies the zero-knowledge proof.

All parameters are passed to the function, as well as the public hash generated from the inputs. This function then checks that the zero-knowledge proof passes and the consumer's hash is eligible in the system.

If the consumer has provided valid information then the zero-knowledge proof passes and the consumer is set-up with an account. This has happened without the user realising, and without the user giving away any private information on the public blockchain.

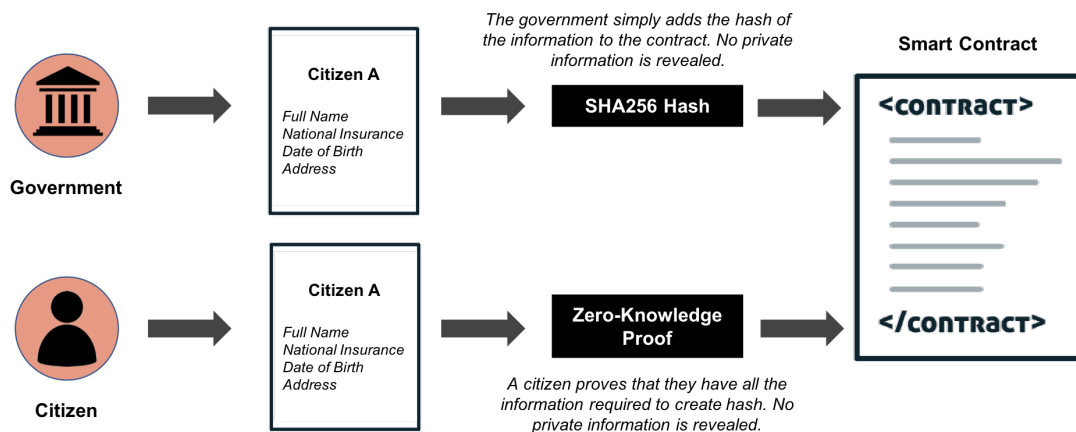


Figure 4.11: WörglCoin's identification system

Chapter 5

Using Wörglcoin

This chapter outlines the React-based web application, with a full user guide detailing how key stakeholders (the contract owner, consumers and businesses) use the service, as well as how the system has been developed so third party applications can accept the UBI token.

5.1 Functionality Overview

The web application is built using the React.js JavaScript library [44] given its ease of use (building individual reusable ‘components’) and its superior rendering performance. Interaction between the front-end and smart contract is done using the web3.js [45] library. The entire system design can be found in Figure 5.1.

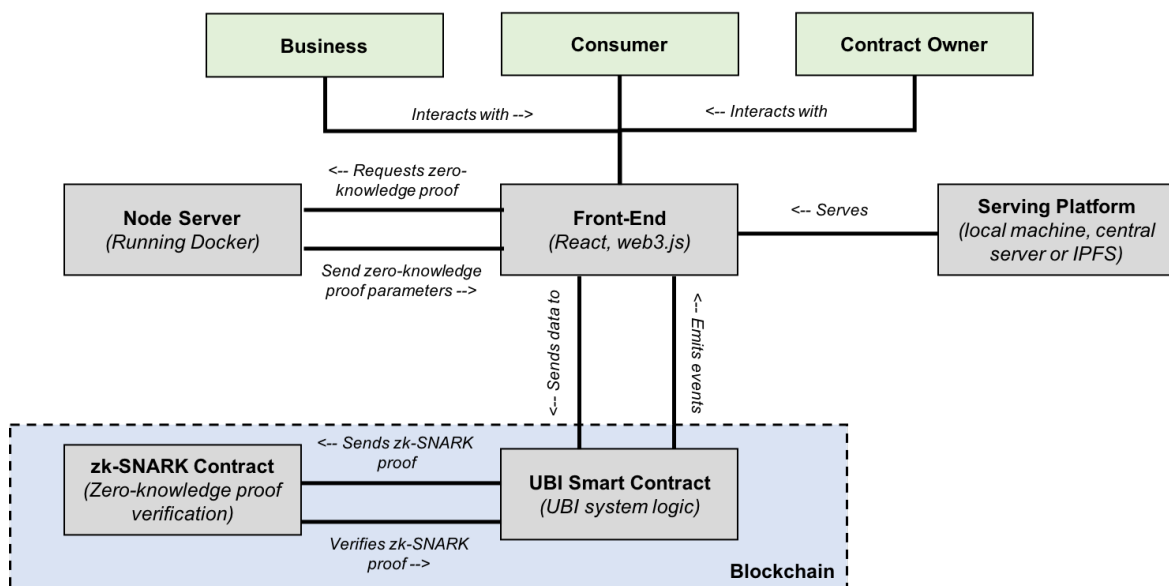


Figure 5.1: The final design of the Wörglcoin system

Installation

Before installing and using the Wörglcoin software, the user needs to have installed: the npm package manager [46], Docker [47] and Metamask [48]. Once these are installed, the following commands can be run to get a local version of the project up and running:

```
$ git clone https://github.com/matthewsmorrison/Worgl-Coin
```

```
$ cd Worgl-Coin
```

```
$ npm install
```

```
$ ganache-cli
```

```
$ truffle migrate
```

```
$ npm run start
```

```
$ node server
```

Using the Software - Contract Owner

Once the website has been hosted on the local machine (the website should be located at <http://localhost:3000/>), navigate to the Metamask extension and ensure that you are connected to the local Ethereum network (see Figure 5.2). Then navigate to your Ganache CLI, copy the private key of the first account (see Figure 5.3) and import it into Metamask (see Figure 5.4).

As the contract owner, you will now be able to navigate to the administration page either by clicking 'Administration' in the top right-hand corner, or navigating to '<http://localhost:3000/administration>'. This screen allows the contract owner to see key application statistics (e.g. how many consumers are registered) as well as interact with the contract (changing the value of the UBI token or resetting the token balance).

Furthermore, this page allows the contract owner to add an individual's information to allow them to sign-up to the service. The owner enters the individual's full name, the national insurance number, the date of birth and the secret phase sent to the individual's address (see Figure 5.5). Once you click 'Add Consumer Hash' this will prompt a Metamask transaction which should be subsequently accepted.

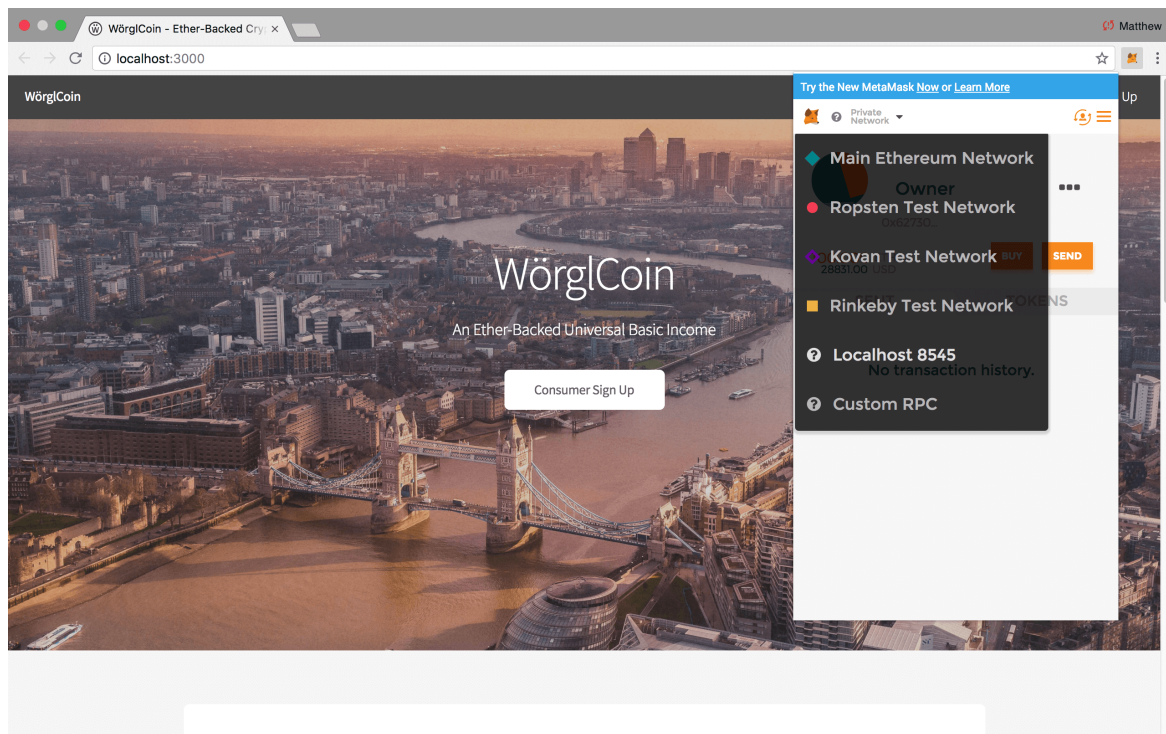


Figure 5.2: Contract owner user guide: step one

```

(dyn3145-144:Worgl-Coin MatthewMorrison$ ganache-cli
Ganache CLI v6.1.6 (ganache-core: 2.1.5)

Available Accounts
=====
(0) 0xda7212641e2c6c632b8d4746fd45f024ba92ccbc (~100 ETH)
(1) 0xca02292fb355640cf13dcdb77431204c4733b66f (~100 ETH)
(2) 0xe386472946af6464cb795da91098b40696262f8a (~100 ETH)
(3) 0xa594fd56984e80c2bec67f1807e6f8dbd99276b7 (~100 ETH)
(4) 0x3b0a5664ac9a077d8536fdbf385b1b0a322411ec (~100 ETH)
(5) 0xdb774fe3ef7c1df8019fae053145ea19ebd0d623 (~100 ETH)
(6) 0x3f530b45cda0c9f68ce70e0a4fff69fd96039ac (~100 ETH)
(7) 0x1310e332c210bda4e1e6bc8627b204b0b0d0bea26 (~100 ETH)
(8) 0x522080d17fb386f5eabc7ea41020ce5f005cfd3e (~100 ETH)
(9) 0x51baf385200d6024682fa1a2f198c1783c4e33 (~100 ETH)

Private Keys
=====
(0) 0x3d002e610416f12c48eb58b6961187b4fd383ff0d867172e2353f07c85ab9762
(1) 0xd6a4a15f35839f1b3483c8efad2f44d4f5b6ab6de81bbe63873b0de7b87ba4f
(2) 0xe06c95c239f59bf8bee500ab1cc599d9a1da4c7321dd63c3d10f3e5c6e7415f7
(3) 0xfe15431ad22a44eb32b8aeeaa021de25a8f875e49276ac3d2b16281bdeb0a789
(4) 0xc01c9acf95bd310ae923a57ff09a569442f94fcd95d5b99a808ea7db842a5eb
(5) 0xf0334aa48272e926b6e929aa82fe2b827f94a51721dfa596da070d05675252f0
(6) 0x3adc87cbc1e09408bd9d80f54359ba3b75f8529d1c4cd59bfcbad6fb1949629
(7) 0x8dd11cf3b16b281b6f38b518df1350e747a74afe0074545e3786ef4b48eed75
(8) 0xdc37adbb0846f639d8b4f628138e2fc22e94abfcb4a5d93b94893399f7ee9108
(9) 0x53462d0ae92ca714ae387d3f2993a8e582fc47e3839ee6969f779a61b1541777

HD Wallet
=====
Mnemonic:      salt aisle cover cannon dance dose speak coach that celery beyond dress
Base HD Path:  m/44'/60'/0'/0/{account_index}

```

Figure 5.3: Contract owner user guide: step two

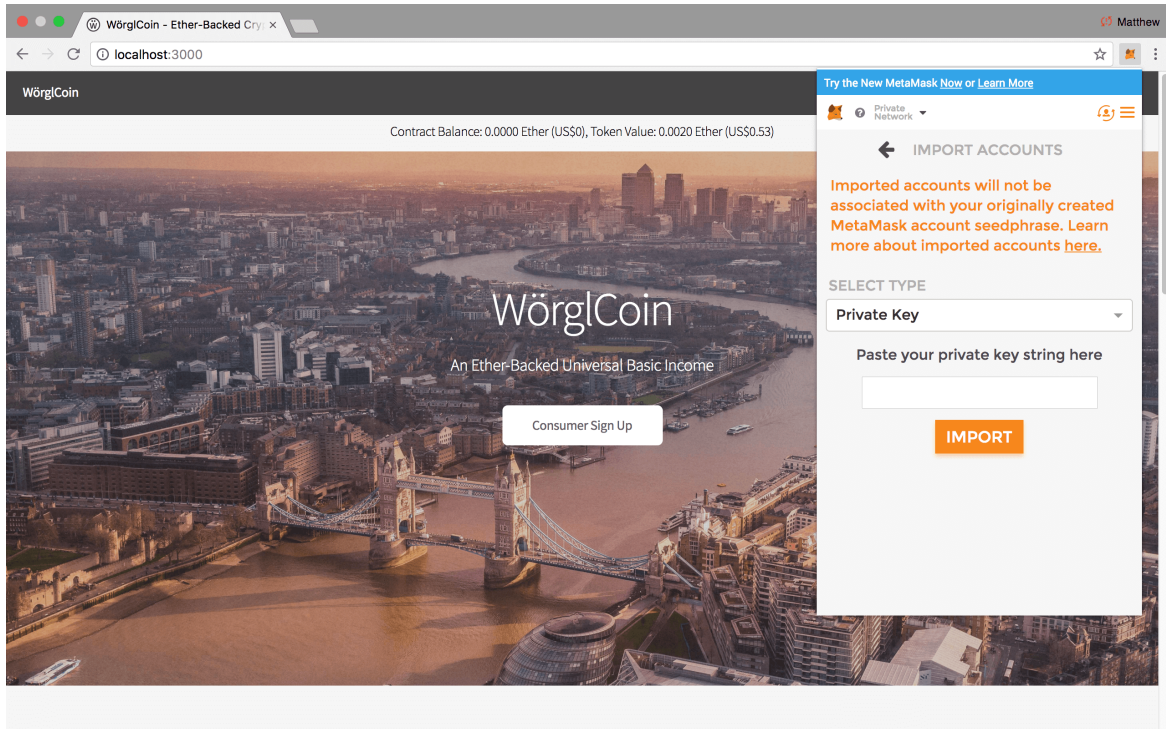


Figure 5.4: Contract owner user guide: step three

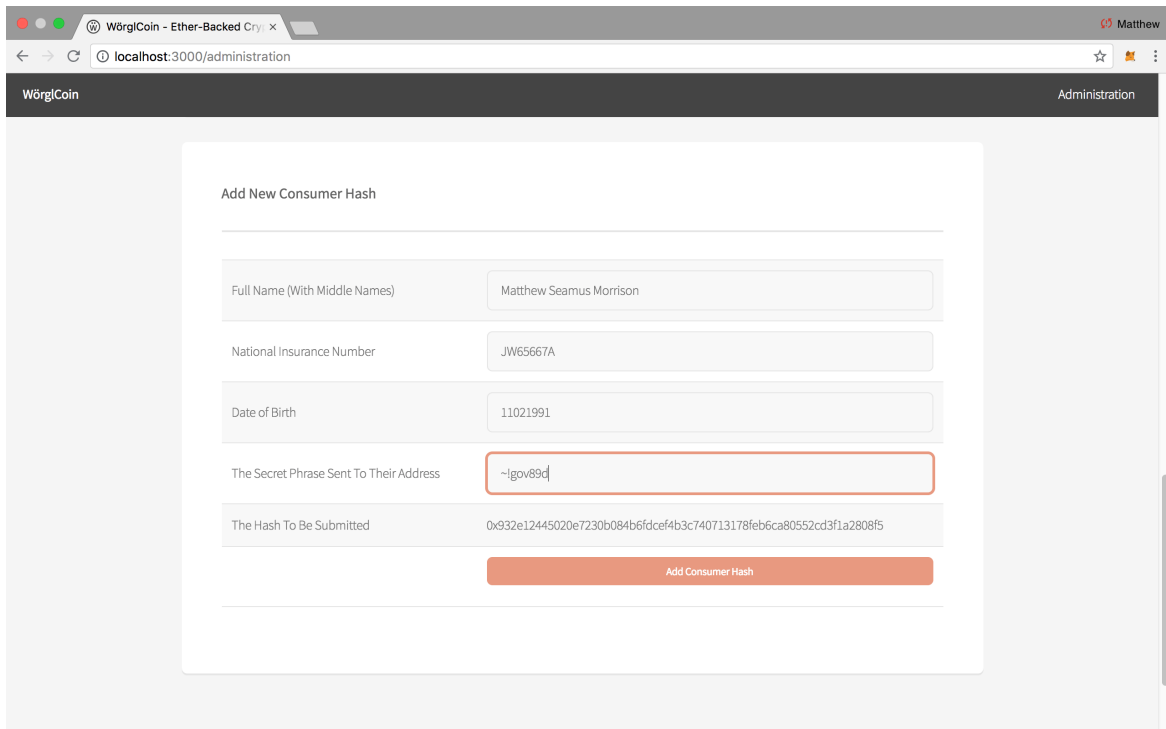


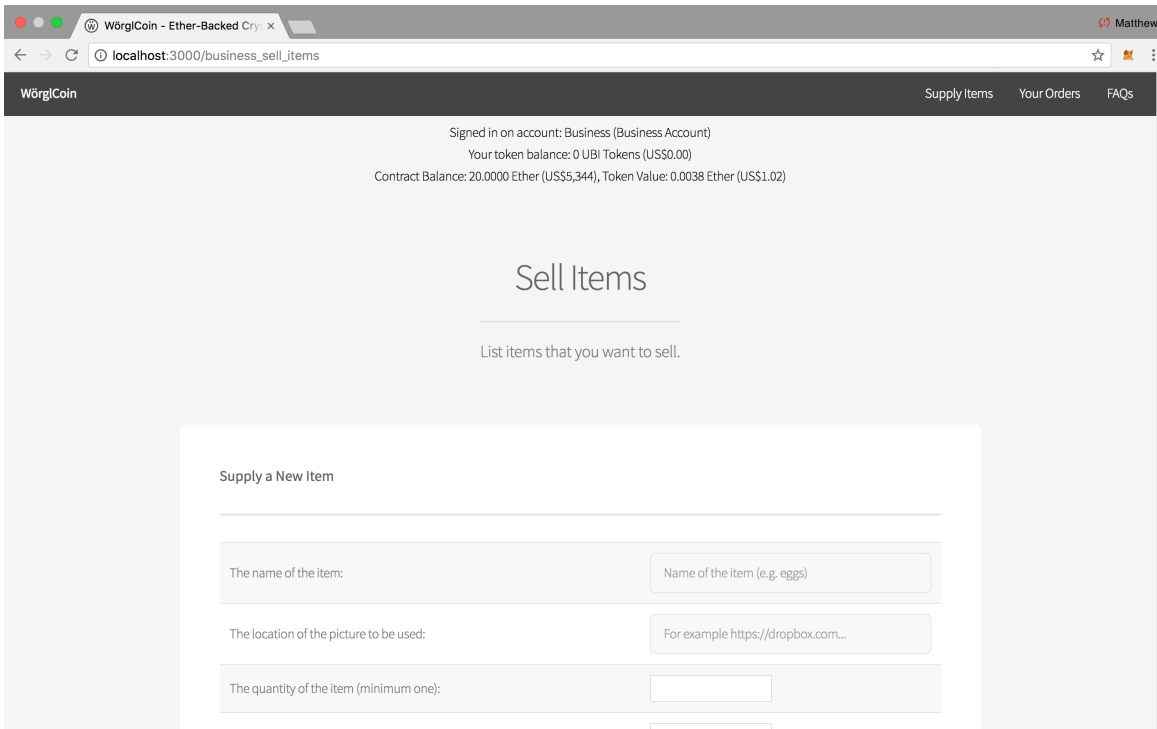
Figure 5.5: Contract owner user guide: step four

Using the Software - Business

Within Metamask, copy the private key of the account that has been added as a business by the contract owner. Once you are logged in as this account, the user interface will change and you will see three options in the navigation panel: 'Supply Items', 'Your Orders' and 'FAQs'. Importantly, the top of the page shows your token balance, the Ether balance of the contract and the value you will get for each UBI token (see Figure 5.6).

As a business, you are able to list items for sale on the site. Navigate to the 'Supply Items' page and enter all the details of the new item (name, URL location of picture, the quantity you have for sale and the price in UBI tokens). Once an item has been added, it will appear on the same page under the heading 'Items Currently Supplied' (see Figure 5.7).

You are also able to see orders that have been received in the 'Your Orders' page. As a business, when an order is received from a consumer, all relevant details are found on this page. You can mark an order as sent once it has been dispatched to the correct delivery location (see Figure 5.8). The item is then located under the heading 'Historical Orders' (see Figure 5.9). A frequently asked questions page can be consulted if there are problems with the application (see Figure 5.10).



The screenshot shows a web browser window with the URL `localhost:3000/business_sell_items`. The page title is "WörglCoin" and the user is logged in as "Business (Business Account)". The account information displayed is: "Your token balance: 0 UBI Tokens (US\$0.00)" and "Contract Balance: 20.0000 Ether (US\$5,344), Token Value: 0.0038 Ether (US\$1.02)". The navigation menu includes "Supply Items", "Your Orders", and "FAQs". The main heading is "Sell Items" with the subtext "List items that you want to sell." Below this is a form titled "Supply a New Item" with three input fields: "The name of the item:" (with placeholder "Name of the item (e.g. eggs)"), "The location of the picture to be used:" (with placeholder "For example https://dropbox.com..."), and "The quantity of the item (minimum one):".

Figure 5.6: Business user guide: step one

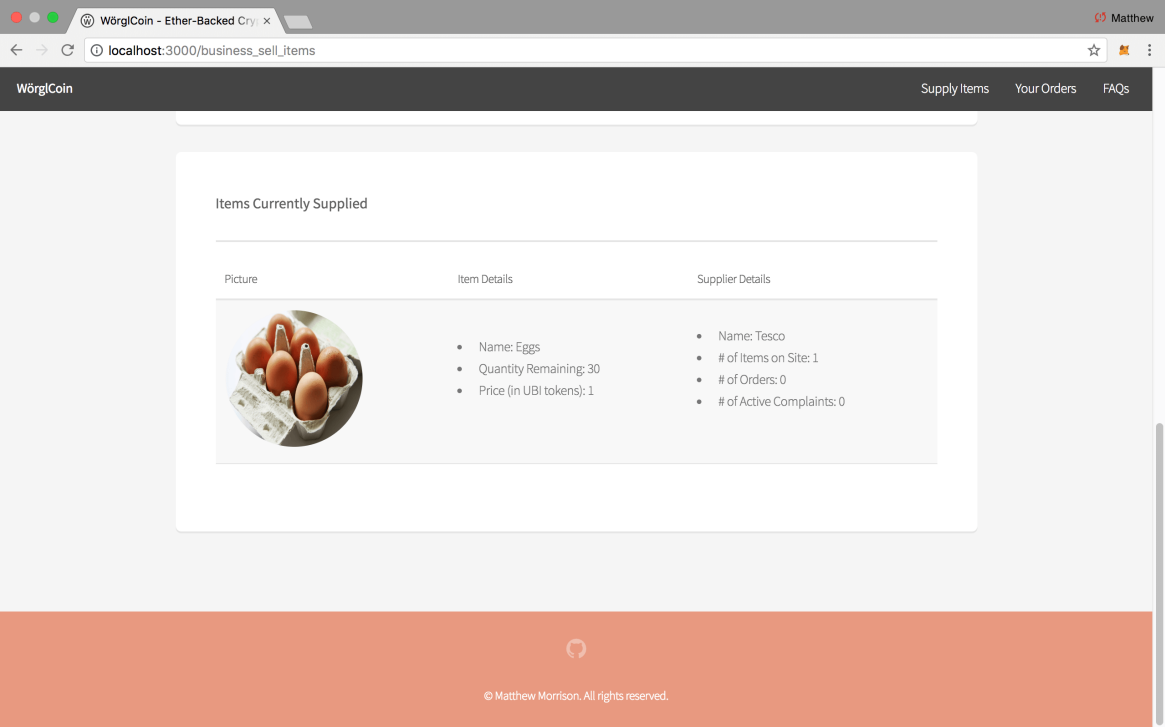


Figure 5.7: Business user guide: step two

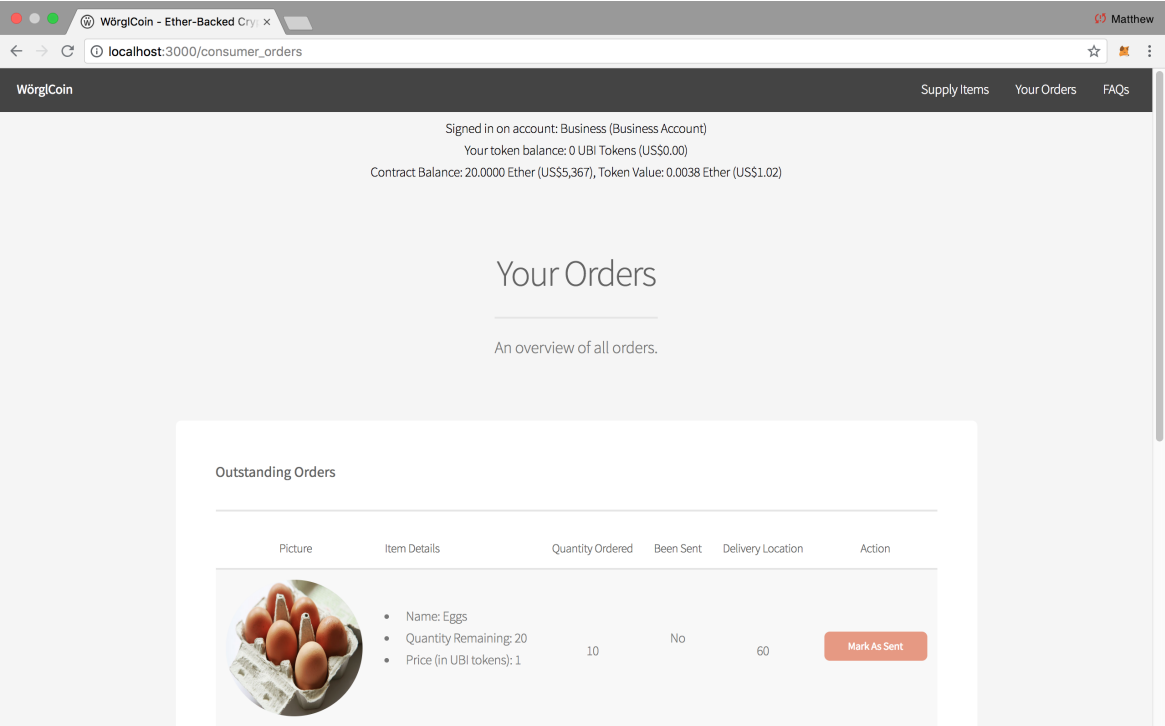


Figure 5.8: Business user guide: step three

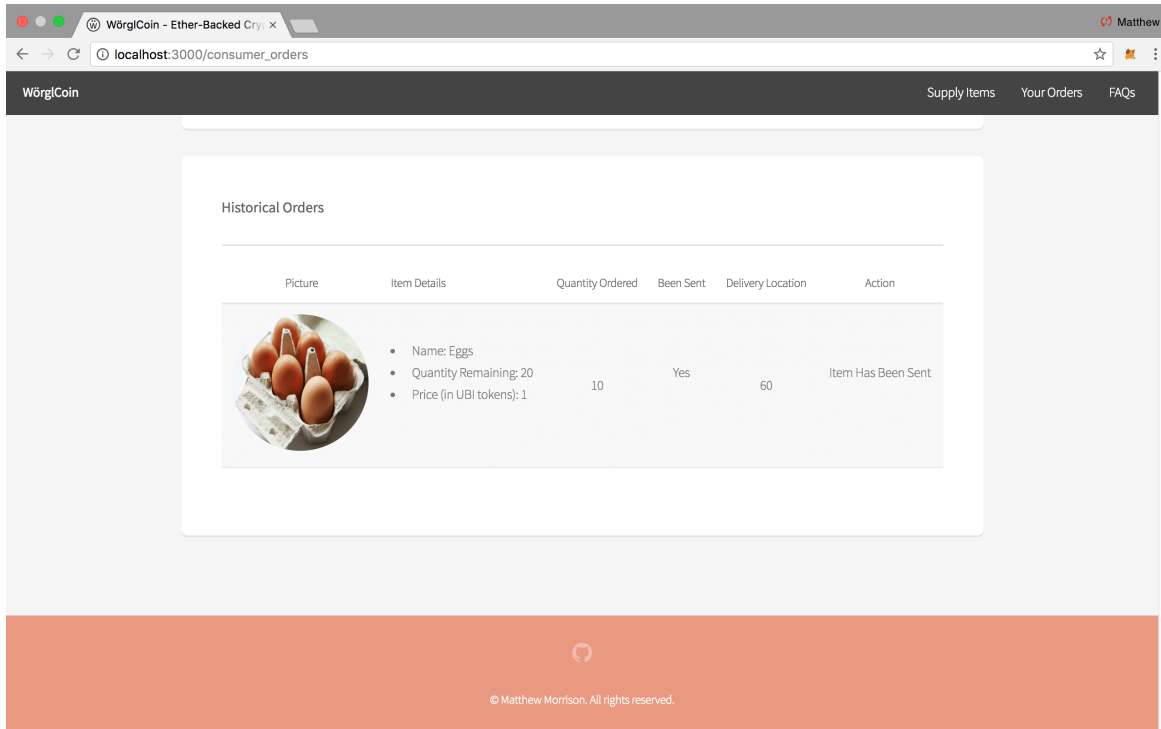


Figure 5.9: Business user guide: step four

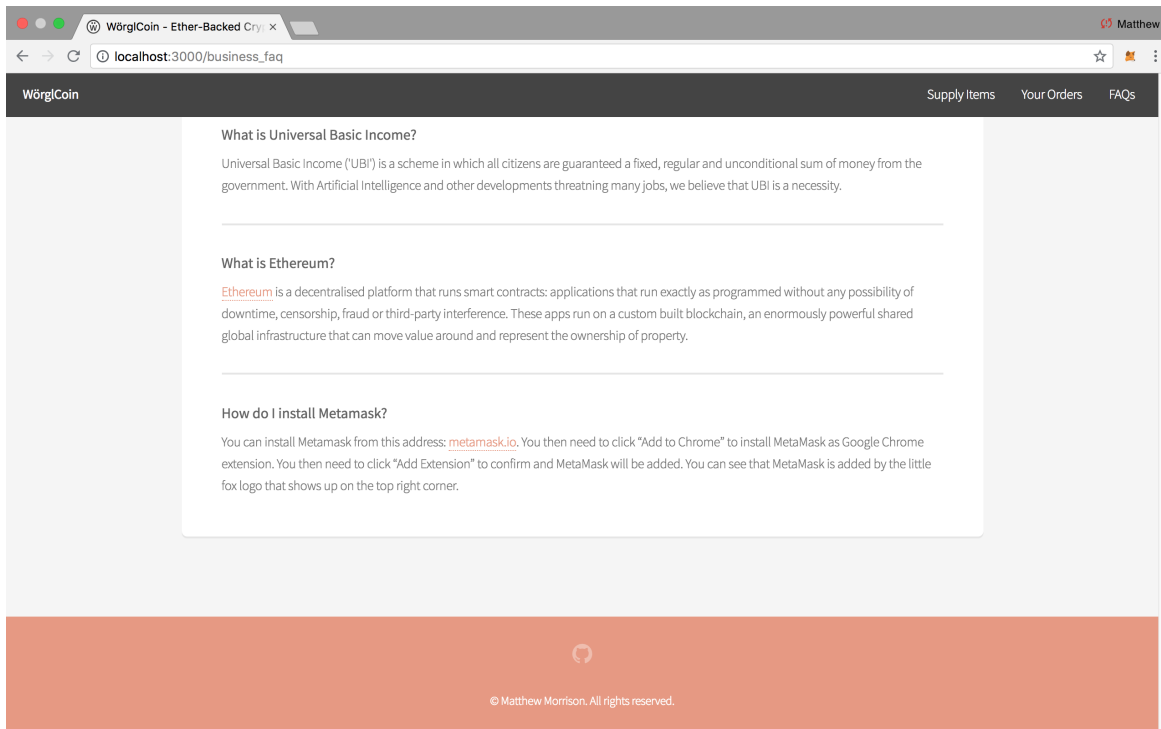


Figure 5.10: Business user guide: step five

Using the Software - Consumer

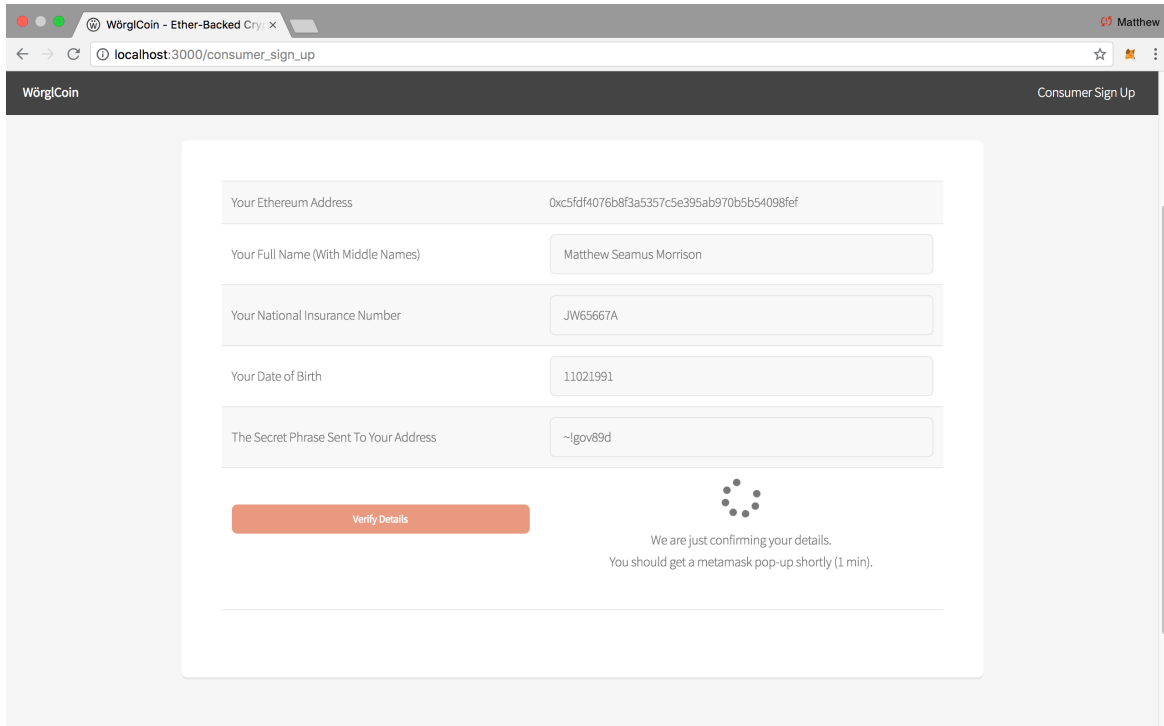
Within Metamask, copy the private key of the account that you want to register as a consumer. Once you are logged in as this account, proceed to the ‘Consumer Sign Up’ page in the top navigation panel. On this page you are able to verify your details and sign-up as a consumer (if the contract owner has also added your details to the site). Type in your details in the correct box and then proceed to click ‘Verify Details’ (see Figure 5.11). It will take around a minute to verify your account and, if successful, you will see a Metamask transaction (which should be accepted) and then a prompt telling you that the account has been registered (see Figure 5.12).

Once you are signed up, you can refresh the page and see your token balance, the contract balance and the token value at the top of the page. As a consumer you can buy items by navigating to the ‘Buy Items’ page. On this page you will see all items for sale and you can then enter the quantity desired and the numeric code of the delivery location for these items. You will be able to see your token balance update in real time as you enter the quantity desired (see Figure 5.13). Click the ‘Buy Item’ button and you will receive a Metamask transaction prompt.

Once an order has been made, you can navigate to the ‘Your Orders’ page and see the current status of your order. If the item has not been marked as sent by the business it will be visible under the ‘Outstanding Orders’ header. You should also notice that your UBI balance has been reduced by the total value of your order at the top of the page (see Figure 5.14).

If an order has been marked as sent by the business, the order will move under the title ‘Historical Orders’. As a consumer, you have the option of making a complaint against the business (see Figure 5.15). A complaint can be made if an item has not been received on time. When a business has a complaint against them, the number of active complaints will be displayed so that other users can evaluate the business. You are also able to reset the complaint once it has been resolved.

There is also a frequently asked questions page that can be used if there are any difficulties. This includes answers to many common problems, including issues with sign-up (see Figure 5.16).



WörglCoin Consumer Sign Up

Your Ethereum Address 0xc5fd4076b8f3a5357c5e395ab970b5b54098fef

Your Full Name (With Middle Names) Matthew Seamus Morrison

Your National Insurance Number JW65667A

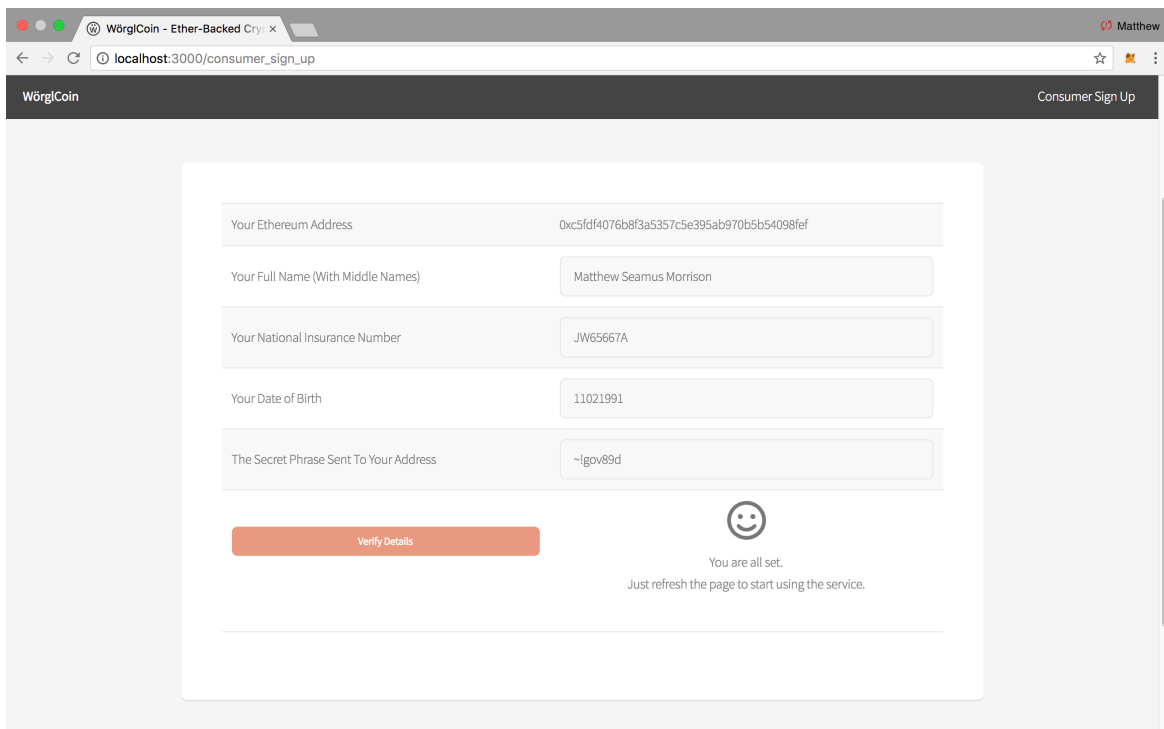
Your Date of Birth 11021991

The Secret Phrase Sent To Your Address ~lgov89d

Verify Details

We are just confirming your details.
You should get a metamask pop-up shortly (1 min).

Figure 5.11: Consumer user guide: step one



WörglCoin Consumer Sign Up

Your Ethereum Address 0xc5fd4076b8f3a5357c5e395ab970b5b54098fef

Your Full Name (With Middle Names) Matthew Seamus Morrison

Your National Insurance Number JW65667A

Your Date of Birth 11021991

The Secret Phrase Sent To Your Address ~lgov89d

Verify Details

You are all set.
Just refresh the page to start using the service.

Figure 5.12: Consumer user guide: step two

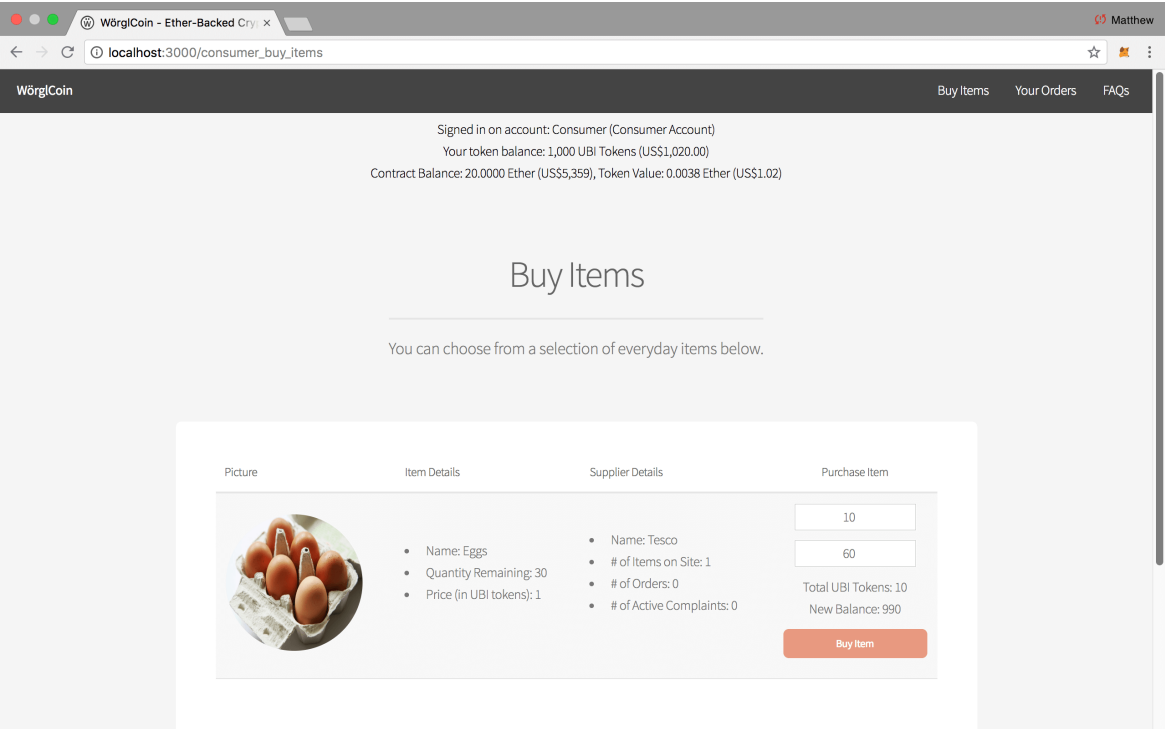


Figure 5.13: Consumer user guide: step three

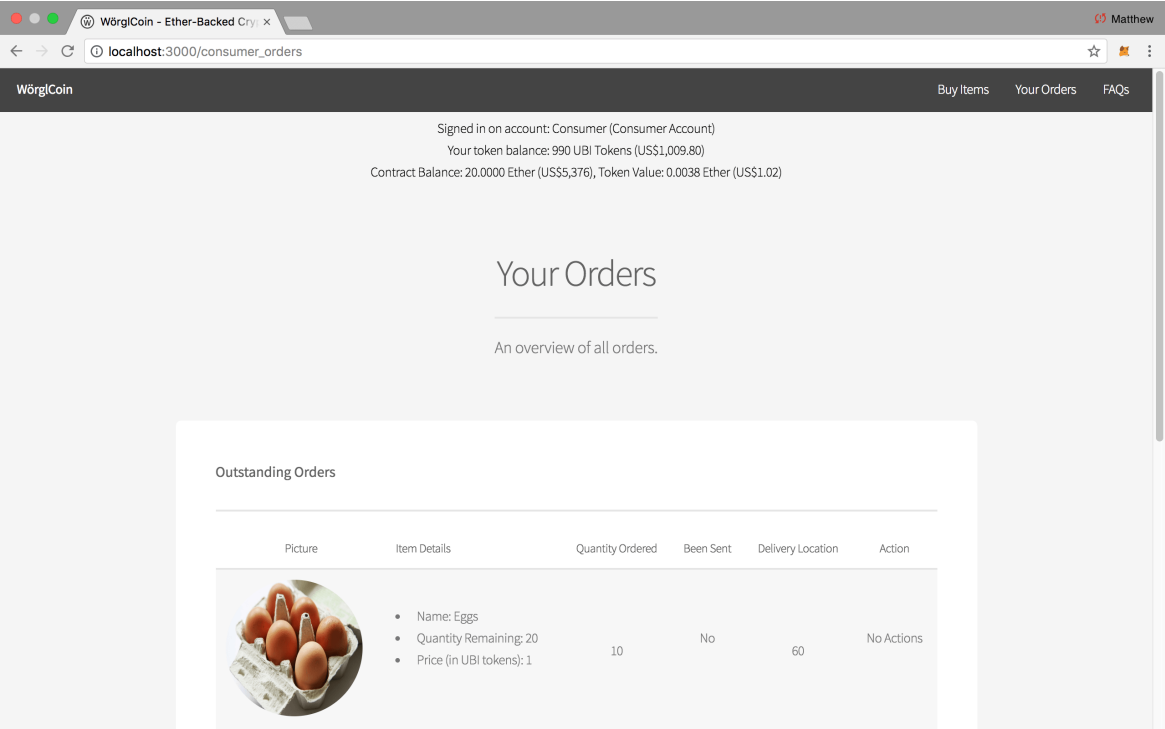


Figure 5.14: Consumer user guide: step four

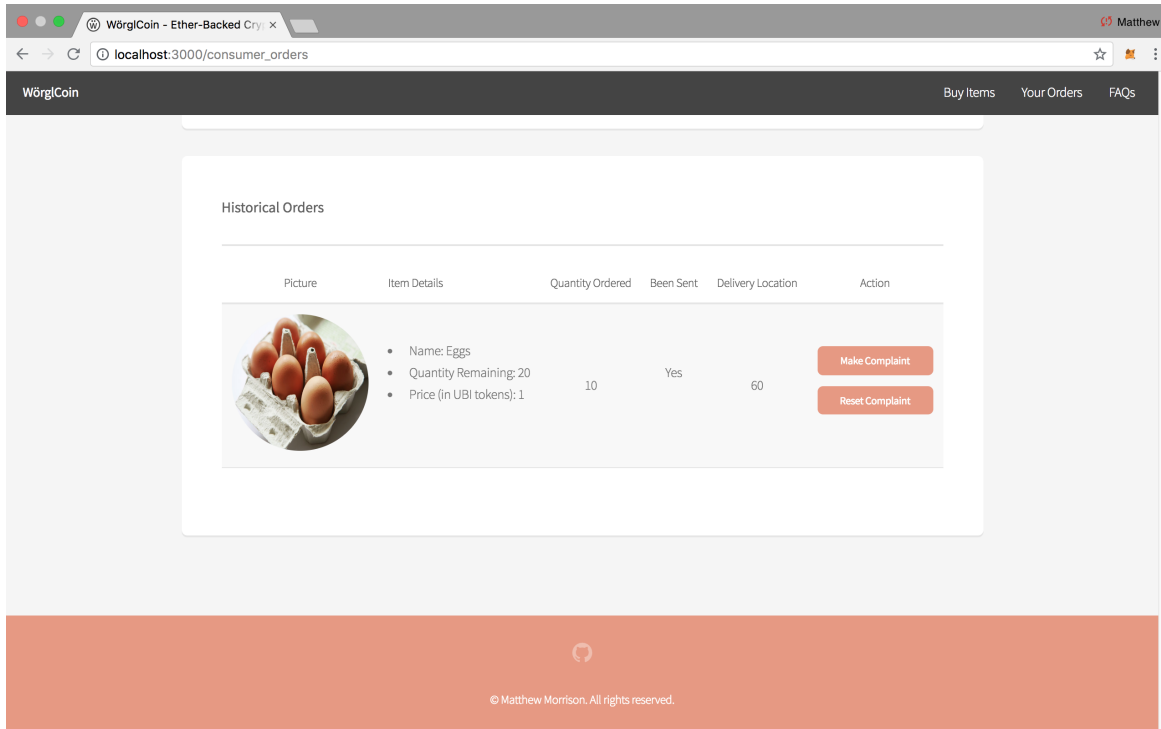


Figure 5.15: Consumer user guide: step five

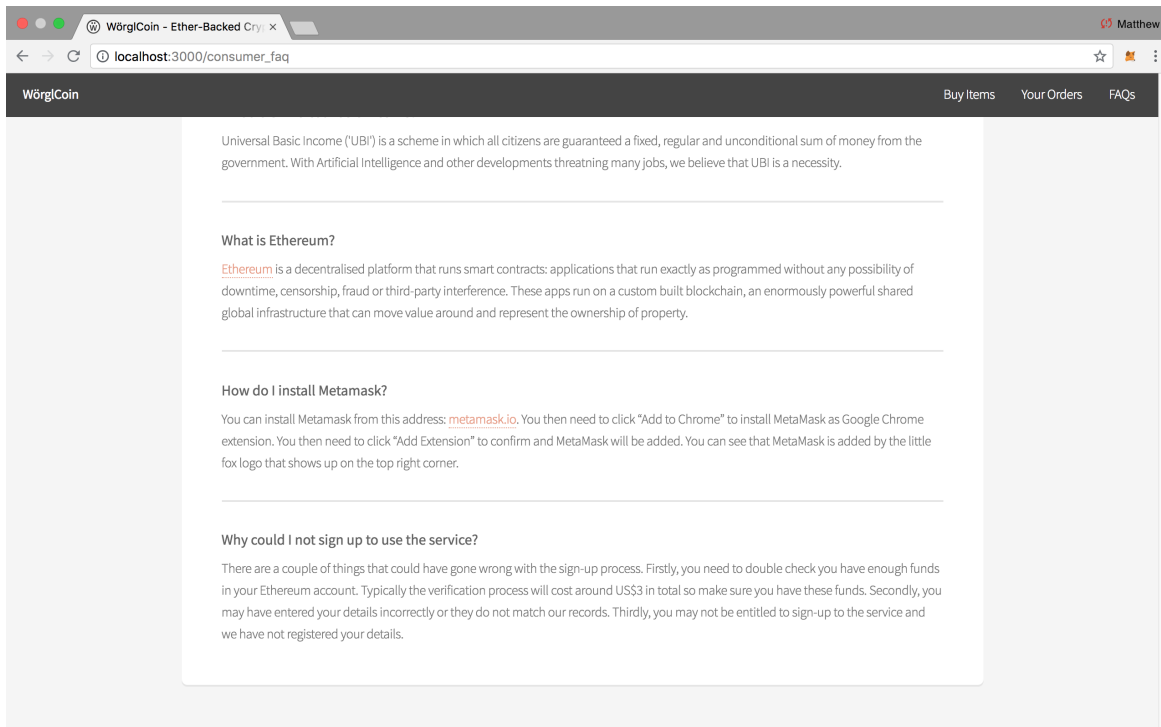


Figure 5.16: Consumer user guide: step six

5.2 UBI Token Interface

Wörglcoin has been developed using a public interface. A third party application can easily access public functions from within the Wörglcoin smart contract to query users' token balances, allow users to buy items and many other actions (see Figure 5.17). By doing this, the Wörglcoin UBI token can be spent on third party websites (assuming the business implementing the application is a registered entity with Wörglcoin) and contribute to the adoption and usability of the token.

A full ERC20 token interface has not been created as Wörglcoin tokens are designed to be non-fungible. This is because consumers should not be able to transfer tokens to one another (as this would undermine the principle of token balances being topped up to a particular level) and businesses should not be able to exchange their tokens with any other party apart from the contract owner.

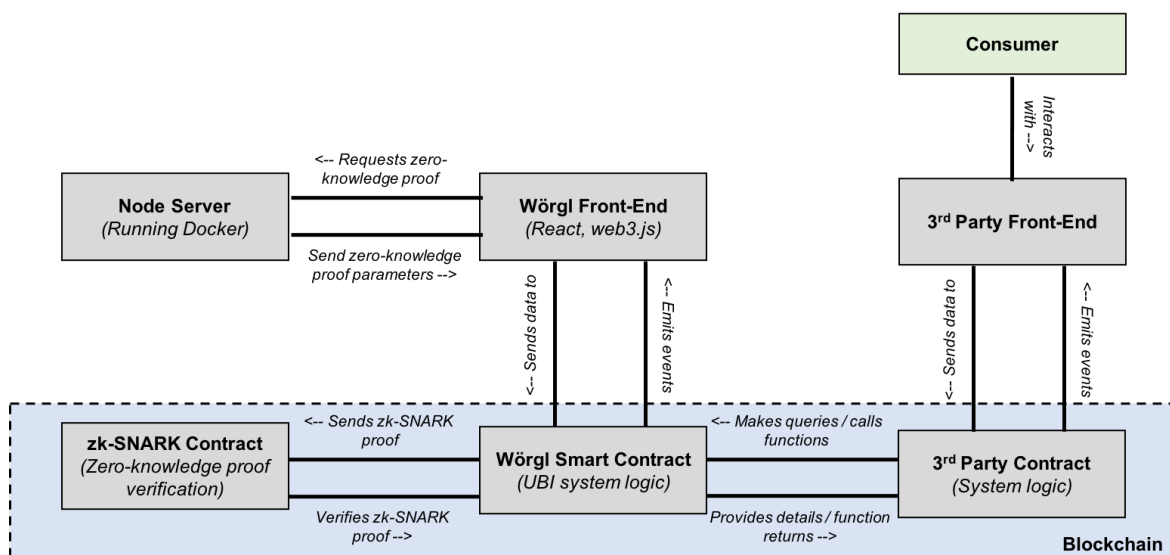


Figure 5.17: The Wörglcoin smart contract interface.

Chapter 6

Evaluation

This chapter outlines the evaluation process of the project. This has been conducted through interviews with Universal Basic Income experts and user testing with potential UBI recipients and merchants.

6.1 Evaluation Methodology

There are four main areas to be evaluated:

1. **Is UBI a credible alternative to the welfare state today?** It is important to get the input of both supporters and non-supporters.
2. **Is blockchain a suitable technology to deliver UBI?** It is important to establish whether blockchain adds important elements lacking in the current payments infrastructure.
3. **Is the use of existing government datasets an effective identification method?** Using existing government data is a new area in the cryptocurrency UBI space. We need to establish whether adopting this approach is sensible and something the government would be willing to implement.
4. **Does backing the token with Ether solve the deflation problem?** As discussed earlier, one of the main problems in cryptocurrency UBI solutions is the de-valuing of the token due to deflationary mechanics. It is important to establish whether the design outlined in this project will achieve the stated goals.

6.2 Interviews

A series of interviews were conducted with experts in the field of Universal Basic Income, from those working on UBI projects to leading think tanks:

- **Alex Howlett from Greshm** [49]. Greshm is a UBI system which uses existing payments infrastructure [50]. The tokens used in the project (XGD tokens) are backed one-for-one by the US dollar, with a mechanism in place managing when an XGD token matures into US dollars. Having been a Software Developer, Alex Howlett is the founder of Greshm and has written a paper on Macroeconomics.
- **Otto Lehto from the Adam Smith Institute** [51]. The Adam Smith Institute is a non-profit and non-partisan think tank based in the UK, typically concentrating on neo-liberal and free-market ideas [52]. The Adam Smith Institute typically promotes the use of a simple welfare system based around a Negative Income Tax or Basic Income that tops up the wages of the poor and guarantees that work always pays. Otto Lehto is currently completing his PhD in Political Economy at Kings College London and co-authored a paper entitled 'Basic Income Around the World - The Unexpected Benefits of Unconditional Cash Transfers' for the Adam Smith Institute [53].
- **Lennard Hulsbos and Paul Anca from C.UBI** [54]. C.UBI is a UBI initiative utilising blockchain technology to enable secure and independent tracing of the world's resources [55]. Lennard Hulsbos and Paul Anca are the founders of C.UBI, with Lennard having focused on bringing sustainable strategies into existing businesses throughout his career and Paul has focused on business, economics and design.
- **Professor Donald Hirsch from the Joseph Rowntree Foundation** [56]. The Joseph Rowntree Foundation is an independent social change organisation working to solve UK poverty [57]. Professor Donald Hirsch is responsible for the Minimum Income Standard (MIS) for the United Kingdom, a programme which researches what income households need to reach a minimum acceptable standard of living in the view of members of the public.

Summarising the feedback from these interviews across the main four topic areas:

Is UBI a credible alternative to the welfare state today? There is mixed opinion on this topic, with some individuals believing that UBI is a long overdue policy (see Figure 6.1 and Figure 6.2) whereas others believe it is entirely infeasible. Arguments against UBI typically centre around the difficulty in convincing a society to unconditionally fund individuals regardless of whether they choose to work or not and the huge expense required to fund the scheme. Furthermore, some question whether a flat payment to each member of a heterogeneous society with different social needs is desirable (see Figure 6.3).

Is blockchain a suitable technology to deliver UBI? Some key principles of an efficient UBI scheme should be transparency (ensuring all citizens know everyone is getting the same amount) and security, so citizens know that they are guaranteed an income in the future (see Figure 6.4). Blockchain technology is a suitable mechanism to deliver the transparency required of such a system. However, some question whether governments are equipped to deal with the complexity of blockchain technology at the current time. Furthermore, there are questions surrounding blockchain's suitability in its current form given its resource intensity and lack of scalability (see Figure 6.5).

Is the use of existing government datasets an effective identification method? The identification method used in this project was cited as a unique and robust way to verify the credentials of an individual, assuming the deployment of this scheme comes from a government entity. Governments already have this data and already uniquely identify individuals, so there is no need to reinvent the wheel (see Figure 6.6). However, some flaws were pointed out, including the fact that a user may be able to sell their credentials to a malicious third party in exchange for fiat currency (see Figure 6.7). However, this was not considered a unique flaw to this particular identification system as all technological solutions are at risk from individuals either selling their credentials or being coerced into doing so. Furthermore, it was mentioned that an identification system, such as the one deployed in this project, would not be required when we move into a world with digital identities which may be government issued.

Does backing the token with Ether solve the deflation problem? It was general consensus that it is difficult to back UBI tokens with real value. The devaluation mechanism used in this project (topping up a user's balance to the same level each month) was cited as a smart way to promote spending of the UBI token (see Figure 6.8). In terms of backing this project's token with Ether, in general it was perceived as a sensible scheme.

“I think UBI is long overdue. Technological innovation and its impact has already been felt. For hundreds of years we have been introducing inefficiencies into the labour market as a way to get money to people. The labour market over time is less and less efficient in terms of allocating labour... Basic income is the answer and, for example, I am not sure the Great Depression would have happened if we had basic income.” [49]

Figure 6.1: Arguments for UBI from Alex Howlett

“I think that I would advocate [UBI] today and I would have advocated it thirty or forty years ago. There may be some resistance and we risk losing some of the key features of UBI, moving towards conditional income or some watered down version of negative income tax. These schemes may restrict the progress of UBI.” [51]

Figure 6.2: Arguments for UBI from Otto Lehto

“The question then arises whether a flat-rate payment to every citizen is the best way to deploy this money. Ultimately it puts all your social eggs in the one basket of giving each person the same amount of money to get by as they choose. But people’s needs are very different, and the social benefits of ensuring services such as education, health and affordable housing may well be considered to merit at least some social resources.” [56]

Figure 6.3: Arguments against UBI from Donald Hirsch

“I think that simplicity and transparency are key features of UBI. They can help generate a basic security for economic agents. They can see [what they will get paid] into the future and monitor and track what their neighbours and other individuals in society are getting paid.” [51]

Figure 6.4: Key features of UBI from Otto Lehto

“I think blockchain in today’s form [is] probably not [suitable]. It is too resource intensive, it’s not scalable, it’s not fast enough. [These things are needed] to deliver a global platform.” [54]

Figure 6.5: Blockchain is not currently ready from Lennard Hulsbos

“A lot of these [UBI] projects are reinventing the wheel. Governments already have ID systems in place. [Your system] is definitely on the right track with that. A lot of these decentralised systems that are trying to create unique identification... will end up reinventing a lot of the institutions they are trying to move away from.” [49]

Figure 6.6: Advantages of identification method from Alex Howlett

“One person could say ‘Hey, I need 500 pounds right now, I am going to sell my credentials to this other person and they can collect my basic income from now on.’ These problems are not specific to your system, there is no way to do this perfectly. Even if you imagine a perfect system with perfect identification, you can come up with a scenario where someone can give their credentials to someone else, or be coerced into doing so. There is no way around that. Any system needs to be robust to some identity fraud.” [49]

Figure 6.7: Criticism of identification method from Alex Howlett

“[Topping up a user’s balance each month] is a really cool idea and I have never heard of anything like that before. I was expecting you to say you have some sort of demurrage system but this is actually much cooler.” [49]

Figure 6.8: Advantages of devaluation mechanism from Alex Howlett

6.3 User Feedback

For further evaluation of this project, twenty individuals from a variety of backgrounds experimented with using the software. The process involved an individual user being sat in front of the software for a period of twenty minutes, with a brief explanation taking place at the beginning covering the purpose of the service and how to sign transactions using Metamask. Following the testing, a user would then be presented with a survey (see Figure 6.9) asking for their feedback on key areas.

In terms of the sample, there was a diverse range of cryptocurrency knowledge (see Figure 6.10). However, there was far more limited knowledge of Universal Basic Income across the participants (see Figure 6.11). Around 40% of test users rated the ease of use of the website five out of five (see Figure 6.12) and 50% of users rated the overall design of the site at five out of five (see Figure 6.13).

The survey also specifically asked users whether the sign up process was intuitive, given the importance of this process to the project overall. Around 75% of users rated the sign up process as four out of five or above (see Figure 6.14), however several comments mentioned that it took a while for the sign up process to complete (see Figure 6.17).

Overall, 70% of users would consider using the service in the future (see Figure 6.15) and 55% would recommend the service to their friends (see Figure 6.16). Overall, the user feedback indicates that the web application simplifies the UBI scheme and significantly hides the back-end complexity.

6.4 Merchant Feedback

In terms of merchant feedback, an interview was conducted with Jonathan Mitchell from the ethical clothing platform 'Brothers We Stand', which has both a physical and online presence [58].

The key feedback from this interview was that cash flow is a primary concern of businesses. Any initiative has to ensure that a business can sustain itself from a cash flow perspective. Therefore, any mechanism that will exchange tokens for value needs to maximise cash flow.

If some form of tax rebate is used (such as a VAT refund), then the discount received needs to be substantial to overcome the cash flow shortage this will cause. Furthermore, a business will only be interested in the service if it increases its user base to a satisfactory level to warrant the increased overhead of accepting tokens.

Finally, the inherent volatility in cryptocurrency today is a concern. A business needs to be secure in the fact that an item is going to be sold for the same amount each month no matter what.

Universal Basic Income

A short survey assessing your views of the Universal Basic Income software you have just used.

***Required**

How familiar are you with cryptocurrency? *

1 2 3 4 5

Not at all Very

How familiar are you with Universal Basic Income? *

1 2 3 4 5

Not at all Very

How easy did you find the website to use? *

1 2 3 4 5

Difficult Easy

How much did you like the design of the website? *

1 2 3 4 5

Not at all Very

How easy did you find the sign up process? *

1 2 3 4 5

Difficult Easy

Would you consider using this service in the future?

Yes

No

Maybe

Would you recommend this service to your friends? *

Yes

No

Maybe

Any suggestions / comments?

Your answer

SUBMIT

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google.
Report Abuse - Terms of Service - Additional Terms

Google Forms

Figure 6.9: Google survey given to users

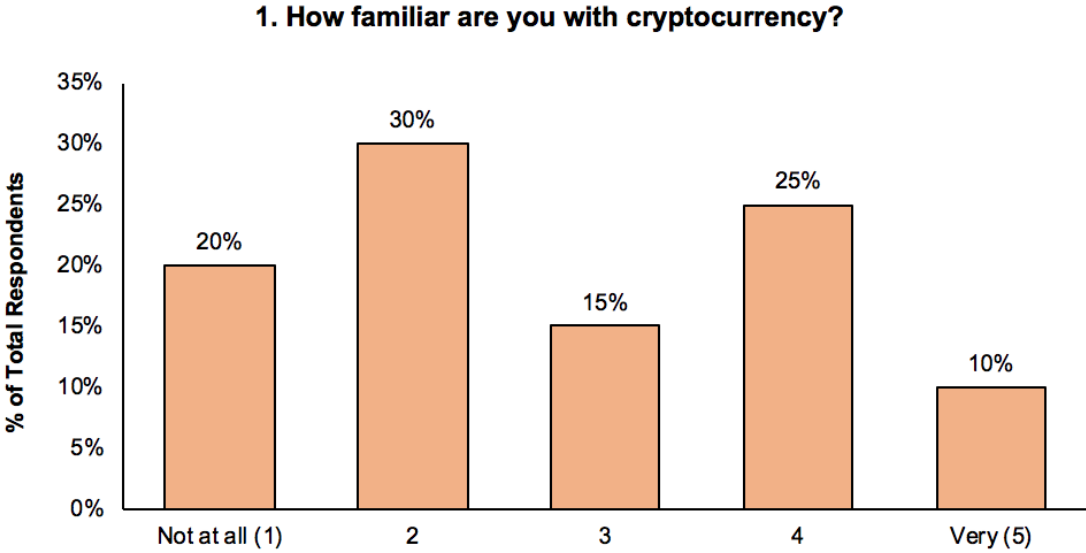


Figure 6.10: Question one of the user survey

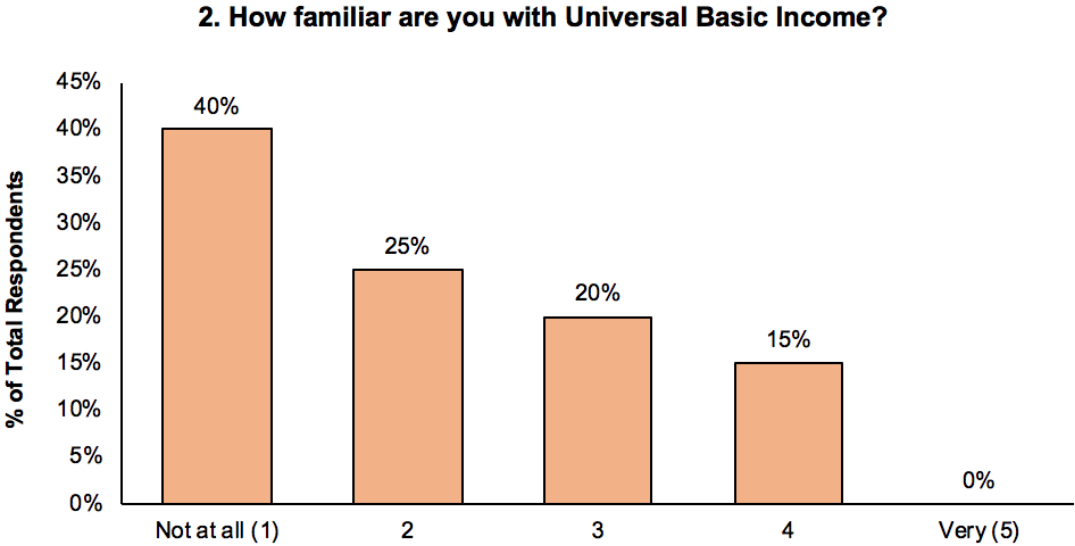


Figure 6.11: Question two of the user survey

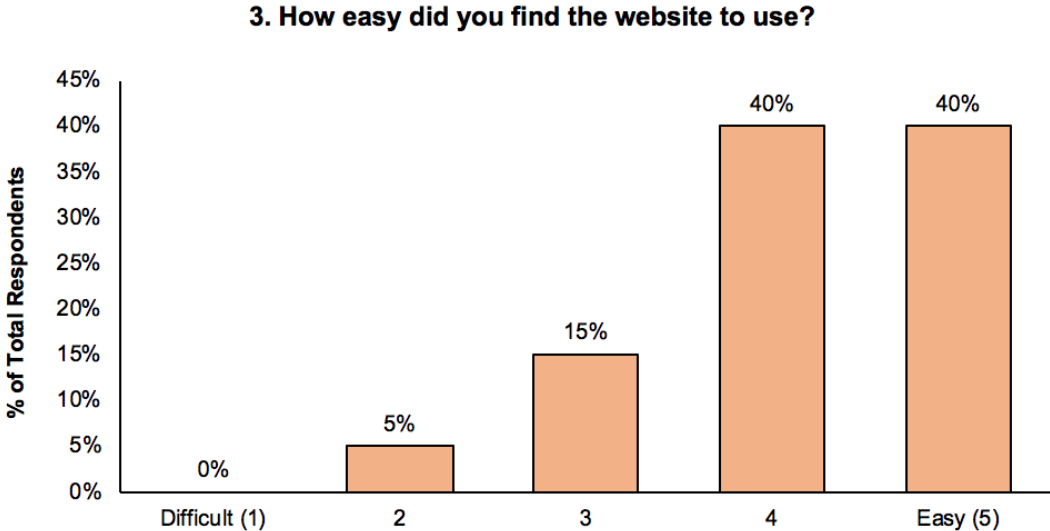


Figure 6.12: Question three of the user survey

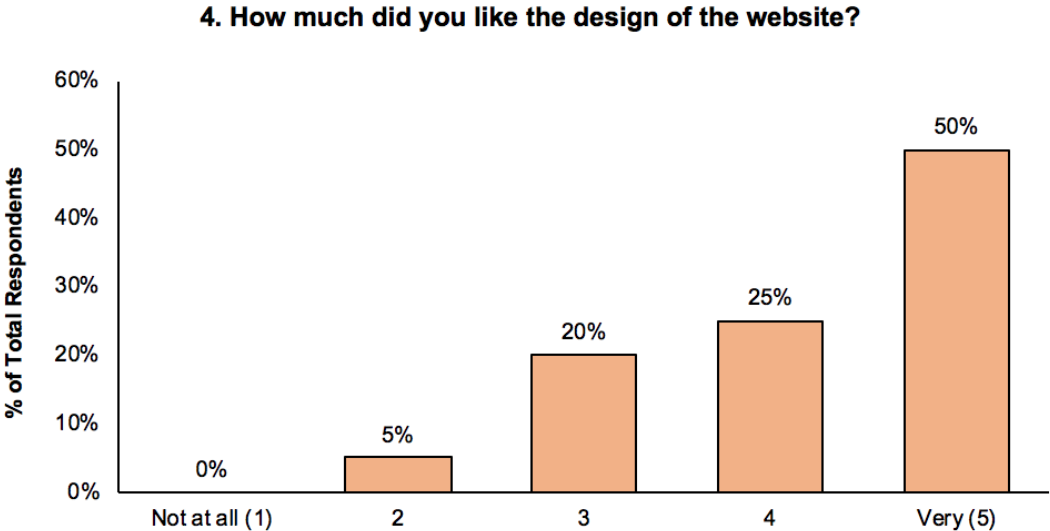


Figure 6.13: Question four of the user survey

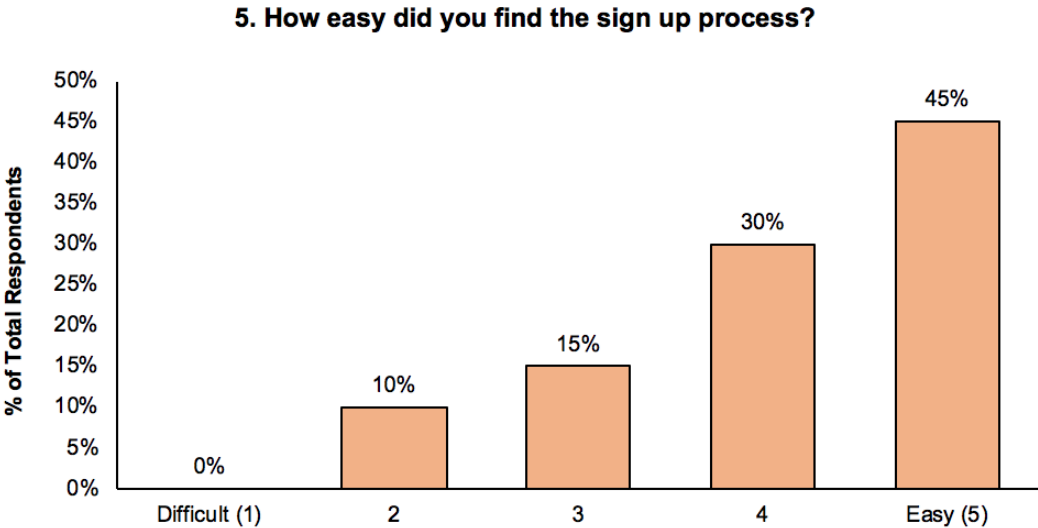


Figure 6.14: Question five of the user survey

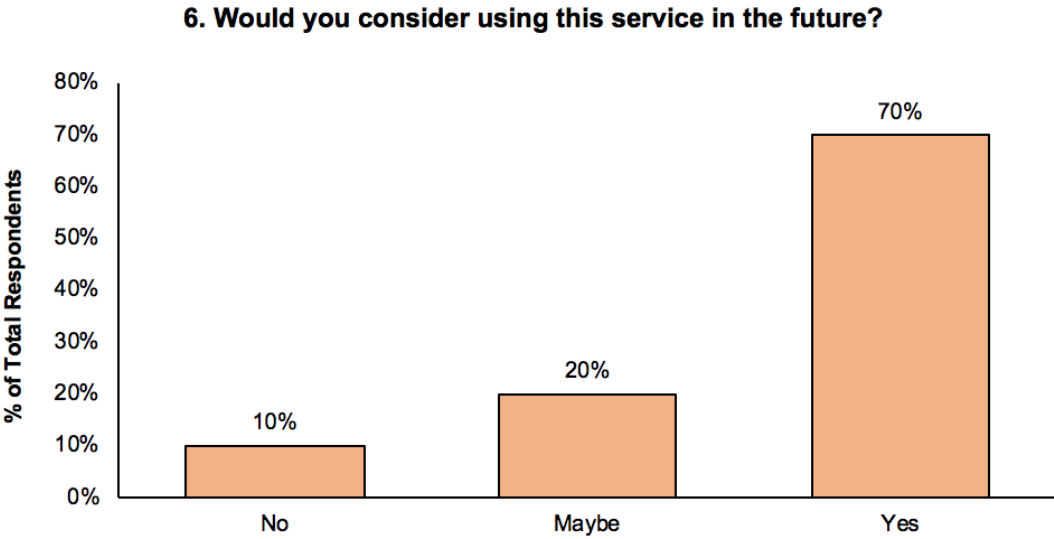


Figure 6.15: Question six of the user survey

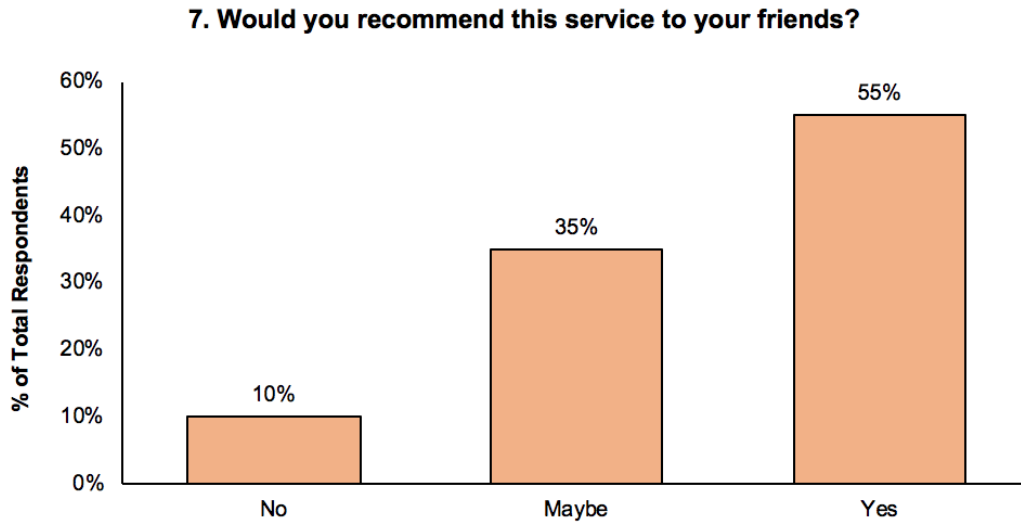


Figure 6.16: Question seven of the user survey

Any suggestions / comments?

7 responses

Great website. Maybe more explanation about what the transactions etc. are for.
The sign up process is too slow and doesn't tell you much.
No
Cool idea
Needs more explanation. The explanation at the beginning only covered basics of transactions but I found them quite confusing.
Why was the sign up process so slow?
UBI is definitely needed. Good implementation.

Figure 6.17: Question eight of the user survey

6.5 Summarising the Feedback

As discussed, the evaluation of this project has been conducted through interviews with experts in the field of Universal Basic Income and user testing with potential UBI recipients and merchants. Summarising across the four main areas:

Is UBI a credible alternative to the welfare state today? There is a significant amount of debate surrounding Universal Basic Income and whether it is an effective policy. Today, a UBI policy would be hard to justify given the huge cost of deploying this scheme and the fact that funds are better directed towards lower income individuals. However, as Artificial Intelligence and other technological innovations disrupt the labour market further, society will find it more difficult to reward labour with stable income. Therefore, in the future, UBI may be the only option for providing a stable source of income.

Is blockchain a suitable technology to deliver UBI? Whilst blockchain is an effective method to deliver Universal Basic Income (providing transparency and flexibility), the cost of the system developed in this project is too high for full-scale adoption across an entire society. Furthermore, with the cost of several transactions being incurred by the consumer (signing up for the service, buying items), this potentially restricts lower income individuals from using the system. The mechanism of incentivising users to spend their tokens is particularly effective and one that is not used in any other system.

Is the use of existing government datasets an effective identification method? The identification method used in this project is a new development in the Universal Basic Income space. The use of pre-existing government data is a robust way of uniquely identifying individuals, with the use of zero-knowledge proofs protecting the privacy of those individuals. However, no system is perfect in the prevention of Sybil attacks. Even users of the UBI scheme developed in this project would be able to sell their credentials to a rogue actor in exchange for fiat currency.

Does backing the token with Ether solve the deflation problem? Whilst the technological mechanism for backing the token with real value is an effective one, the use of cryptocurrency and its current volatility will not be attractive for merchants using the site. The value of the largest cryptocurrencies can vary significantly month-on-month, meaning that: 1) merchants have to accept this volatility or 2) the contract owner needs to constantly change the exchange rate for the token. Neither of these solutions are ideal.

6.6 Addressing the Feedback

Based on the evaluation of this project a number of improvements could be made:

Is Blockchain a suitable technology to deliver UBI? The use of a side-chain or different blockchain technology (potentially utilising proof-of-authority rather than proof-of-work) could bring down the transaction costs of this system.

Is the use of existing government datasets an effective identification method? A secondary layer of protection needs to be added to the identification method in this project. A mechanism by which rogue accounts can be reported and assessed by humans is required, or a technological solution of using Artificial Intelligence to identify rogue accounts. This would add another layer of protection against Sybil attacks.

Does backing the token with Ether solve the deflation problem? In order to reduce the volatility of the value received by merchants, a stable coin or fiat currency needs to be used. A method of plugging in the smart contracts to existing payment infrastructure could be developed, or the use of a stable coin. This would provide a more stable source of income to merchants and make them more likely to accept the token for goods and services.

Chapter 7

Conclusion

This chapter summarises the achievements that have been made in this project and examines future potential areas of research in cryptocurrency-based UBI systems and zero-knowledge proofs on Ethereum.

7.1 Summary of Achievements

This project addressed the universe of UBI schemes, their current drawbacks and proposed a new UBI system backed by Ether, a devaluation mechanism to incentivise spending and a zero-knowledge proof based system for uniquely identifying individuals.

An Ethereum-based, smart contract system has been built to deploy a UBI token and all the logic underpinning the commercial use of the token. Furthermore, the system allows businesses that have received UBI tokens to exchange them for Ether at a pre-defined rate as determined by the contract owner. This ensures that businesses are incentivised to list real products and services for sale as the token is underpinned by real value.

We have thoroughly reviewed security principles and measures implemented to mitigate these risks. Furthermore, given that one of the most important aspects of a government-backed UBI system is cost, the system has been built with efficiency in mind, with a number of techniques implemented to minimise the cost of deployment and function execution.

Zero-knowledge proofs on Ethereum have been explored and a proof-of-concept system has been implemented using new, in-development techniques. Furthermore, a novel approach has been taken to verify the pre-image of a SHA-256 hash. This system allows users to verify their data and check it against existing government-held data. Importantly, users do not need to publish any private information on the Ethereum blockchain.

7.2 Evaluation Overview

As outlined in Chapter 6, a thorough evaluation process has been conducted using a number of interviews with experts in the field of Universal Basic Income and user testing with potential UBI recipients and merchants. The key evaluation points that arose:

1. Is UBI a credible alternative to the welfare state today?

- **Summary:** there is a significant amount of debate surrounding Universal Basic Income and whether it is an effective policy. Today, a UBI policy would be hard to justify given the huge cost of deploying this scheme, and the fact that funds are better directed towards lower income individuals. However, as Artificial Intelligence and other technological innovations disrupt the labour market further, society will find it more difficult to reward labour with stable income. Therefore, in the future, UBI may be the only option for providing a stable source of income.

2. Is blockchain a suitable technology to deliver UBI?

- **Summary:** whilst blockchain is an effective method to deliver Universal Basic Income (providing transparency and flexibility), the cost of the system developed in this project is too high for full-scale adoption across an entire society. Furthermore, with the cost of several transactions being incurred by the consumer (signing up for the service, buying items), this potentially restricts lower income individuals from using the system. The mechanism of incentivising users to spend their tokens is particularly effective and one that is not used in any other system.
- **Improvements:** the use of a side-chain or different blockchain technology (potentially utilising proof-of-authority rather than proof-of-work) could bring down the transaction costs of this system.

3. Is the use of existing government datasets an effective identification method?

- **Summary:** the identification method used in this project is a new development in the Universal Basic Income space. The use of pre-existing government data is a robust way of uniquely identifying individuals, with the use of zero-knowledge proofs protecting the privacy of those individuals. However, no system is perfect in the prevention of Sybil attacks. Even users of the UBI scheme developed in this project would be able to sell their credentials to a rogue actor in exchange for fiat currency.
- **Improvements:** a secondary layer of protection needs to be added to the identification method in this project. A mechanism by which rogue accounts can be reported and assessed by humans is required, or a technological solution of using Artificial Intelligence to identify rogue accounts. This would add another layer of protection against Sybil attacks.

4. Does backing the token with Ether solve the deflation problem?

- **Summary:** whilst the technological mechanism for backing the token with real value is an effective one, the use of cryptocurrency and its current volatility will not be attractive for merchants using the site. The value of the largest cryptocurrencies can vary significantly month-on-month, meaning that: 1) merchants have to accept this volatility or 2) the contract owner needs to constantly change the exchange rate for the token. Neither of these solutions are ideal.
- **Improvements:** in order to reduce the volatility of the value received by merchants, a stable coin or fiat currency needs to be used. A method of plugging in the smart contracts to existing payment infrastructure could be developed, or the use of a stable coin. This would provide a more stable source of income to merchants and make them more likely to accept the token for goods and services.

7.3 Applications

With the growing interest in Universal Basic Income, the techniques that have been used in this project could benefit a number of different parties.

Other UBI Projects: a number of other UBI projects explored in Chapter 2 of this report are concerned with the value of their token and how to uniquely identify users in their system. Based on the system implemented in this project, these systems can explore new methods to cover these concerns.

Government: across the globe there has been growing political backing for a UBI scheme. However, a lot of the pilots being run are expensive to implement and offer no flexibility. A government could explore the system in this project and propose a low-cost, flexible way to implement their very own UBI scheme. Furthermore, simple pilots could be run using an off-the-shelf deployment of this system.

Academia: researchers with an interest in Universal Basic Income, blockchain and cryptocurrency could benefit from exploring the system that has been implemented in this project. Furthermore, this system could be built upon depending on the particular area of interest for the researcher (e.g. making the token fungible and freely tradeable).

Developers: given the novel implementation of a SHA-256 pre-image zero-knowledge proof using ZoKrates and Docker, the techniques explored in this project can be used by other developers to create proof-of-concepts for their applications.

7.4 Future Work

There are a number of areas which can be further explored beyond the findings presented in this project.

Optimising Zero-Knowledge Proofs on Ethereum

As it stands, the zero-knowledge proof identification system that has been implemented in this project is fairly costly. In November 2017, Ethereum introduced the Byzantium update which added a number of cryptographic primitives to their smart contracts. The smart contracts used for zero-knowledge proof verification could be potentially improved by using these cryptographic primitives. By reducing the cost of verification, it makes the system much more feasible and affordable for potential users.

More Secure Zero-Knowledge Proof System

As it stands, the system implemented in this project uses a zero-knowledge proof that verifies the first 32 bits of the hash produced by the SHA-256 compression function. This has been done as it would be too costly to verify a full 256 bit hash on the Ethereum blockchain today. Therefore, future work can be done to explore how to verify a full 256-bit hash on-chain using a number of potential techniques, including packing the bits into a number of output fields and creating new data types. Furthermore, the functionality of ZoKrates should be expanded to allow a full SHA-256 hash function to be utilised.

Developing Third-Party Applications on Top of UBI Coin

An interface has been created for the UBI token so that other businesses can allow signed-up users to spend tokens on their site. Future work could concentrate on how to make this system more open and actually develop third-party applications on top of the UBI system.

Digital Fiat Currency

The system designed in this project has been backed using Ether, the cryptocurrency behind the Ethereum network. However, the ideal backing for such a system is a digital version of Fiat currency, such as a Digital Pound or a Digital Dollar. Depending on the evolution in this space, future work could be done around backing this system with a digital version of Fiat currency or a stable coin which is backed by Fiat currencies.

Outsourced Delivery System

The delivery system outlined in this project is fairly primitive; a numerical code is assigned to delivery locations and businesses know these locations. Future work could concentrate on how to outsource the delivery mechanism to an established company with a national or international presence. This would require creating a way for the delivery company to know the delivery address of the consumer without broadcasting these details.

Bibliography

- [1] CNBC LLC. *Elon Musk: Robots will take your jobs, government will have to pay your wage*. [Video] 2016. Available from: <https://www.cnbc.com/video/2016/11/04/elon-musk-robots-will-take-your-jobs-government-will-have-to-pay-your-wage.html> [Accessed 23rd January 2018].
- [2] Global News. *Facebook CEO Mark Zuckerberg delivers Harvard commencement full speech*. [Video] 2017. Available from: <https://www.youtube.com/watch?v=4VwElW7SbLA> [Accessed 18th August 2018].
- [3] More T. *Utopia*. Bedford/St. Martin's, Boston, Massachusetts, 1999.
- [4] Vives JL. *On Assistance to the Poor*. University of Toronto Press, Toronto, Canada, 1999.
- [5] Wikimedia Foundation, Inc. *Thomas More*. Available from: https://en.wikipedia.org/wiki/Thomas_More [Accessed 23rd August 2018].
- [6] Wikimedia Foundation, Inc. *Johannes Ludovicus Vives*. Available from: https://commons.wikimedia.org/wiki/File:Johannes_Ludovicus_Vives.jpg [Accessed 23rd August 2018].
- [7] Brundage A. *The English Poor Laws*. Palgrave Macmillan, Basingstoke, United Kingdom, 2002.
- [8] Wellcome Collection. *Poor Law Amendment Act*. Available from: <https://google.com/images/zh6ZQH> [Accessed 23rd August 2018].
- [9] Cunliffe J, Erreygers G. *The Enigmatic Legacy of Charles Fourier: Joseph Charlier and Basic Income*. Duke University Press, Durham, North Carolina, 2001.
- [10] Friedman M. *Capitalism and Freedom*. University of Chicago Press, Chicago, Illinois, 1962.
- [11] King ML. *Where Do We Go From Here: Chaos or Community*. Beacon Press, Boston, Massachusetts, 1967.
- [12] Office for Budget Responsibility. *An OBR guide to welfare spending*. Available from: http://obr.uk/docs/dlm_uploads/An-OBR-guide-to-welfare-spending-March-2018.pdf [Accessed 11th August 2018]. March 2018.

- [13] World Inequality Lab. *World Inequality Report 2018*. Available from: <https://wir2018.wid.world/files/download/wir2018-summary-english.pdf> [Accessed 11th August 2018]. December 2017.
- [14] International Monetary Fund. *World Economic Outlook*. Available from: <http://www.imf.org/en/Publications/WEO/Issues/2017/04/04/world-economic-outlook-april-2017> [Accessed 11th August 2018]. April 2017.
- [15] PricewaterhouseCoopers LLP. *Will robots really steal our jobs?*. Available from: <https://www.pwc.co.uk/economic-services/assets/international-impact-of-automation-feb-2018.pdf> [Accessed 30th May 2018]. February 2018.
- [16] Quartz. *Bill Gates: We should tax the robot that takes your job*. [Video] 2017. Available from: <https://www.youtube.com/watch?v=nccryZ0crUg> [Accessed 26th January 2018].
- [17] DALIA RESEARCH GmbH. *The EU's Growing Support for Basic Income*. Available from: <https://basicincome.org/wp-content/uploads/2017/05/DR-2017-survey.p/> [Accessed 4th June 2018]. May 2017.
- [18] Opinion Desk at The Independent. John mcdonnell's universal basic income idea is interesting but will need a lot more thought. *The Independent*. 31st July 2018. Available from: <https://www.independent.co.uk/voices/editorials/john-mcdonnell-labour-corbyn-universal-basic-income-practicalities-needs-more-thought-a8471596.html> [Accessed 11th August 2018].
- [19] Kansaneläkelaitos - The Social Insurance Institution of Finland. *Objectives and implementation of the Basic Income Experiment*. Available from: <http://www.kela.fi/web/en/basic-income-objectives-and-implementation> [Accessed 15th May 2018].
- [20] Y Combinator Research, Inc. *The first study of basic income in the United States*. Available from: <https://basicincome.ycr.org> [Accessed 15th May 2018].
- [21] Mannabase, Inc. *Manna Whitepaper*. Available from: <http://www.grantcoin.org/documents/manna-whitepaper.pdf> [Accessed 20th May 2018].
- [22] Ethereum Foundation. *A Next-Generation Smart Contract and Decentralized Application Platform*. Available from: <https://github.com/ethereum/wiki/wiki/White-Paper> [Accessed 20th January 2018].
- [23] Ethereum Foundation. *Solidity Documentation*. Available from: <https://solidity.readthedocs.io/en/develop/index.html> [Accessed 20th April 2018].
- [24] Buterin V. *Vyper Documentation*. Available from: <https://vyper.readthedocs.io/en/latest> [Accessed 28th June 2018].

- [25] Edgington B. *Documentation for the LLL compiler*. Available from: <http://111-docs.readthedocs.io/en/latest> [Accessed 28th June 2018].
- [26] Shrans F. *Writing Safe Smart Contracts in Flint*. Master's thesis. Imperial College London; 2018.
- [27] Siegel D. Understanding the dao attack. *Coindesk*. 25th June 2016. Available from: <https://www.coindesk.com/understanding-dao-hack-journalists/> [Accessed 11th July 2018].
- [28] Manning A. *Solidity Security: Comprehensive list of known attack vectors and common anti-patterns*, . Weblog. Available from: <https://blog.sigmaprime.io/solidity-security.html> [Accessed 13th July 2018].
- [29] OpenZeppelin. *SafeMath Library*. (Version 1.12.0) [Code] Zeppelin. Available from: <https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/math/SafeMath.sol>. 2018.
- [30] SC-Forks. *solidity-coverage*. (Version 0.5.11) [Software] npm. Available from: <https://www.npmjs.com/package/solidity-coverage>. 2018.
- [31] National Institute of Standards and Technology. *Digital Identity Guidelines*. 800-63B. Colorado: US Department of Commerce; 2018.
- [32] Zetter K. Hackers clone e-passports. *Wired*. 8th March 2006. Available from: <https://www.wired.com/2006/08/hackers-clone-e-passports> [Accessed 4th August 2018].
- [33] Big Foundation. *Basic Income Guarantee Protocol*, . Available from: <https://github.com/bigfoundation/Documentation/blob/master/BIGwhitepaperEN.md> [Accessed 30th May 2018].
- [34] Facebook, Inc. *Q3 2017 Quarterly Report*. 2017. Available from: https://s21.q4cdn.com/399680738/files/doc_financials/2017/Q3/Q3-17-Earnings-call-transcript.pdf [Accessed 4th August 2018].
- [35] Quisquater JJ. et al. How to explain zero-knowledge protocols to your children. In: Brassard G (ed.) *Advances in Cryptology - CRYPTO 89 Proceedings*. Springer, New York; 1990. p.628-631.
- [36] Bitansky N, Canetti R, Chiesa A, Tromer E. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Simons Institute for the Theory of Computing: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. New York: ACM; 2012. p.326-349.
- [37] Buterin V. *Quadratic Arithmetic Programs: from Zero to Hero*. Weblog. Available from: <https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649> [Accessed 5th August 2018].

- [38] Sullivan N. A *(Relatively Easy To Understand) Primer on Elliptic Curve Cryptography*. Weblog. Available from: <https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography> [Accessed 6th August 2018].
- [39] Buterin V. *Exploring Elliptic Curve Pairings*. Weblog. Available from: <https://medium.com/@VitalikButerin/exploring-elliptic-curve-pairings-c73c1864e627> [Accessed 5th August 2018].
- [40] Buterin V. *Zk-SNARKs: Under the Hood*. Weblog. Available from: <https://medium.com/@VitalikButerin/zk-snarks-under-the-hood-b33151a013f6> [Accessed 5th August 2018].
- [41] Bellare M, Palacio A. *The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols*, . Available from: <https://www.iacr.org/archive/crypto2004/31520273/bp.pdf> [Accessed 25th August 2018].
- [42] SCIPR Lab. *libsnark*. [Software] SCIPR Lab. Available from: <https://github.com/scipr-lab/libsnark>. 2018.
- [43] Eberhardt J. *ZoKrates*. [Software] ZoKrates. Available from: <https://github.com/JacobEberhardt/ZoKrates>. 2018.
- [44] Facebook, Inc. *React*. (Version 16.4) [Software] Facebook, Inc. Available from: <https://reactjs.org>. 2018.
- [45] ConsenSys LLC. *Web3*. (Version 0.18.4) [Software] npm. Available from: <https://github.com/ethereum/web3.js>. 2018.
- [46] Schlueter IZ. *npm*. [Software] npm. Available from: <https://www.npmjs.com/get-npm>. 2018.
- [47] Docker, Inc. *Docker*. (Version 18.03.1) [Software] Docker, Inc. Available from: <https://www.docker.com/get-started>. 2018.
- [48] ConsenSys LLC. *MetaMask*. (Version 4.9.3) [Software] ConsenSys LLC. Available from: <https://metamask.io>. 2018.
- [49] Howlett A. Interviewed by: Morrison M. 21st August 2018.
- [50] Howlett A. *Project Greshm*. Available from: <http://www.greshm.org/files/greshm.pdf> [Accessed 19th August 2018].
- [51] Lehto O. Interviewed by: Morrison M. 30th August 2018.
- [52] Adam Smith Institute. *About the Adam Smith Institute*. Available from: <https://www.adamsmith.org/about-the-asi> [Accessed 19th August 2018].
- [53] Lehto O. *Basic Income Around The World: The Unexpected Benefits of Unconditional Cash Transfers*. ASI (Research) Ltd, London, United Kingdom, 2018.

-
- [54] Hulsbos L, Anca P. Interviewed by: Morrison M. 30th August 2018.
- [55] Hulsbos L, Anca P. *CircularUBI*. Available from: <http://www.circularubi.org> [Accessed 20th August 2018].
- [56] Hirsch D. Interviewed by: Morrison M. 23rd August 2018.
- [57] Joseph Rowntree Foundation. *About the Joseph Rowntree Foundation*, . Available from: <https://www.jrf.org.uk/about-us> [Accessed 20th August 2018].
- [58] Brothers We Stand. *About Brothers We Stand*. Available from: <https://www.brotherswestand.com> [Accessed 29th August 2018].

Glossary

A summary of the key terms used in this report can be found below.

Artificial Intelligence (AI): the development of systems that are able to perform tasks typically associated with human intelligence, including visual recognition, decision-making, among many others.

Basic Income Guarantee (BIG): another term for Universal Basic Income.

Blockchain: a digital database in which data is recorded chronologically in a verifiable and permanent way.

Cryptocurrency: a digital or virtual currency that uses cryptography and blockchain technology.

Decentralised Borderless Voluntary Nation (DBVN): a virtual nation with all laws and regulations hardcoded into smart contracts written on a blockchain.

Ethereum: a decentralised platform that runs smart contracts. These smart contracts run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.

Ethereum Virtual Machine ('EVM'): a run-time environment for smart contracts on the Ethereum network.

Flint: a new type-safe, capabilities-secure, contract-oriented programming language designed for writing robust smart contracts on Ethereum.

Low-level Lisp-like Language ('LLL'): one of the original Ethereum smart contract programming languages and it is a low level language similar to Assembly.

Negative Income Tax (NIT): a progressive income tax system where people earning below a certain amount receive supplemental pay from the government instead of paying taxes. Very similar to Universal Basic Income with only a few minor differences.

Smart Contracts: a section of executable code that run exactly as programmed on

the blockchain without any possibility of downtime, censorship, fraud or third-party interference.

Solidity: a high-level language that developers use to build decentralised applications. Solidity code is compiled into EVM byte-code and executed across every node in the network.

Sybil Attack: where malicious users create fraudulent accounts in order to subvert and impede the functionality of your system.

Universal Basic Income (UBI): a periodic cash payment unconditionally delivered to all participants, regardless of income or any other factor.

Vyper: a smart contract development language built to be secure and simple. Certain functionality has been removed when compared to Solidity including modifiers and function overloading.

Zero-Knowledge Proof: a cryptographic construct (or protocol) that allows one party ('Alice') to prove to another party ('Bob') that they know a public value 'x', without revealing any private information that has led to them knowing 'x'.

zk-SNARK: a non-interactive zero-knowledge proof which is succinct (the messages are very small in comparison to the size of the computation required) and the prover will not be able to construct a proof/argument without knowing a so-called witness.

Appendix A

Ethics Checklist

Below can be found the LSEPI Checklist for this project summarising some of the main ethical considerations:

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?	✓	
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from Human Embryos/Foetuses i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓

Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	✓	
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?	✓	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
Section 10: LEGAL ISSUES		

Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
Section 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Appendix B

Legal, Social and Ethical Considerations

Given the nature of this project and its direct implication for government policy, it is extremely important to consider all of the legal, social and ethical considerations. Whilst this project simply explored a theoretical implementation of such a government-backed scheme, there are still a number of considerations that should be made for full implementation, all of which have been thoroughly researched.

Given that a full deployment of this software would involve human participants giving over sensitive personal data (e.g. National Insurance numbers) a significant amount of focus needs to be placed on the security and proper use of this data. The system has been designed in such a way that a user has full control over their personal data and, by using zero-knowledge proofs, this information is never broadcast to the public Ethereum blockchain.

Secondly, the environmental impact of Bitcoin has been widely criticised, given its large energy use in the mining process. Given that the same criticism can be made of the Ethereum network (which has been utilised in the creation of our UBI coin), we need to be aware of the potential negative environmental impact of such a scheme and the perceptions around that.

Thirdly, there are some moral considerations to be made with respect to the government potentially having granular access to the spending habits of its citizens. As such, it is important to have a clear ethical framework in place before the implementation of any scheme such as the one proposed in this paper.

Several interviews and user testing took place within this project. All participants were between the ages of 25 and 50 (no children or minors) and were chosen from a sample of individuals living in London and attending Imperial College London. An oral statement was given by participants granting consent to use all results and comments given in this report. If permission was not granted then the user or interviewee was not included.

Appendix C

Non-Cryptocurrency UBI Experiments

Country/Organisation	Brief Description	Years	Yearly Value (US\$)	Number of Participants ('000)
USA and Canada	Took place across various states in the US and areas in Canada. Results of the scheme were ambiguous.	1960-1980	Various	~8-9
Alaska	Alaska distributes an unconditional yearly dividend to its residents linked to oil revenues. Has ranged in value over the years.	1982-today	~200-2,000	~630
Namibia	Organised by various NGOs and church groups, the execution was poor and no scientifically valid results can be concluded.	2008-2009	~120	~1
Iran	A monthly dividend paid to residents based on oil revenues of the state. Not a serious UBI scheme given the low level of income received.	2010-today	~600	~50,000-72,000
India	Organised by UNICEF, a number of villages were given small cash transfers. Control groups were used. These experiments saw a rise in living conditions, nutrition and health.	2010-2011	~60	~6
Kenya	A privately funded UBI experiment. Of the 26k participants, only 6k will be paid across the entire experiment.	2016-2027	~300	~26
Uganda	The charity 'Eight' are funding a two year UBI scheme in the region of Fort Portal.	2017-2019	~240	~0.2
Finland	A random sample of 2,000 unemployed people aged 25 to 58 are being paid ~€580 per month by the Finnish government, with no requirement to seek or find employment.	2017-2019	~8,000	~2
Y Combinator	Recruited ~3,000 people across two US states. Randomly assign 1,000 participants to receive US\$1,000 per month for 3-5 years. The remaining 2,000 will be used as a control group.	2017-2022	~12,000	~2-3
Canada	A program that will pay ~4,000 participants between the ages of 18 and 64. The participants will be a mix of those in low-paying jobs and those on social assistance. Amount decreases with work.	2017-2020	~13,000	~4
Scotland	Trials have received support from Nicola Sturgeon and a £250k grant has been given to support the project. Details have not been announced.	2018-unknown	Unknown	Unknown
Netherlands	Participants will be divided into four groups, each of which will receive payments under different conditions. The pilot program has not received support from government as of writing.	Unknown	Unknown	Unknown

Figure C.1: Summary of non-cryptocurrency UBI experiments

Appendix D

Cryptocurrency UBI Solutions

Name	Website	ERC20 Token	Identification Process	Crypto/Asset/Fiat Backed	Currency Distributed
Altrui.st	https://altrui.st/	✗	✗	✗	Monero
BIG Foundation	http://big.foundation/	✓	✓	✗	BIG
Bitnation	https://tse.bitnation.co/	✓	✗	✗	XPAT
Cicada	http://iamcicada.com/	✗	✓	✗	Unknown
Circles	https://joincircles.net/	✗	✗	✗	Individual
Cubecoin	https://cubecoin.net/	✗	✓	✗	CUB
C.UBI	http://www.circularubi.org/	Unknown	Unknown	Unknown	Unknown
Democracy Earth	https://www.democracy.earth/	✓	✗	✗	VOTE
Duniter	https://duniter.org/en/	✗	✗	✗	Ĝ1
Enumivo	https://www.enumivo.org/	✗	✗	✓	ENU UBI Token
Frink	https://frink.global/	✗	✗	✗	Frinks
Grantcoin	http://www.grantcoin.org/	✗	✗	✗	Grantcoin (GRT)
Greshm	http://www.greshm.org/	✗	✓	✓	XGD
Group Income	https://groupincome.org/	Unknown	Unknown	Unknown	Unknown
Manna	https://www.mannabase.com/	✗	✗	✗	MANNA
Project UBU	https://www.projectubu.com/	✓	✓	✗	UBU and UBX
Resilience Project	http://www.resilience.me/	Unknown	Unknown	Unknown	Unknown
Solidar	https://solidar.it/	✗	✗	✗	SDR
Steem Basic Income	https://steemit.com/@steembasicincome	✗	✗	✗	SBI
SwiftDemand	https://www.swiftdemand.com/	✗	✓	✗	Swifts
The Kuwa Foundation	http://www.kuwa.org/	✗	✓	✗	Unknown
UBIC	http://www.ubicoin.com/	✗	✓	✓	BIQT
Ubit	http://www.unibit.io/	✗	✗	✗	UBIT

Figure D.1: Summary of cryptocurrency UBI solutions