A digital identity system based on a basic feature phone

Cian Michael Murphy

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Engineering (Computer Engineering)

Supervisor: Donal O'Mahony

April 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

[illegible]

Cian Michael Murphy

April 19, 2022

Permission to Lend and/or Copy

[illegible]

Cian Michael Murphy

April 19, 2022

A digital identity system based on a basic feature phone

Cian Michael Murphy, Master of Engineering
University of Dublin, Trinity College, 2022

Supervisor: Donal O'Mahony

Proof of identification is necessary to access basic services, such as opening a bank account, but in 2018 almost a billion people did not have an official proof of identity. Most of these people live in underdeveloped countries. More than 80% of people without formal identity come from sub-saharan Africa and South East Asia(43). At the same time, the concept of decentralised digital "self-sovereign identity" (SSI) has been gaining popularity since Bitcoin launched in 2008.However, most examples of digital identity in the world today require a smartphone to use. This further limits people from underdeveloped countries from obtaining an identity even digitally. This paper proposes a system using USSD, a text-based telecommunications protocol, and Ethereum-based, decentralised identity smart contracts to create a digital identity system which can be accessed by people in countries with less smartphone availability, and presents a proof-of-concept application which illustrates how the system would work for users.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The idea behind this project came about when discussing the EU Covid-19 vaccination certificates, which were built using technology normally associated with decentralised digital identity and self-sovereign identity (SSI). These digital certificates have been essential for accessing bars and restaurants and travelling within the EU throughout 2021 and 2022 and have generally proven to be quite a successful tool. However, one issue with them is that the certificates are only available to people who own a smartphone or other smart device. They can also be printed on a piece of paper, but at least a smart phone is required to verify the certificate.

Although this is not a major issue in Europe where the majority of people own smartphones, it could be an issue for people in developing countries who do not have access to a smart device. This paper explores a potential solution to this issue by proposing a system which allows users to add, present, and remove claims about their identity (e.g. their vaccination status) using just a basic feature phone without internet connectivity. The proposed solution uses a combination of a blockchain-based identity system and an Unstructured Supplementary Service Data (USSD) interface, which allows users to interact with the blockchain without a direct Internet connection.

After introducing and discussing some of the key concepts in the field of digital and decentralised identity, this paper will then propose a solution that allows individuals to control their digital identity without a smart device or direct connection to the Internet, through the use of an Unstructured Supplementary Service Data (USSD) interface and a feature phone. The article will primarily focus on the use of blockchain technology to create a digital identity system that allows individuals to add, remove, and present claims, which are signed and verified by an issuing body, and will discuss the positives and negatives of using blockchain for such a system.

Following this, USSD will be introduced and presented using examples such as M-

Pesa, a mobile money application used in Kenya which also uses USSD to enable digital transactions in an environment where many people do not have access to traditional bank accounts. The proposed application combines these two technologies to create a service through which users can manage their digital identity with just a basic feature phone and a GSM (Global System for Mobile Communication) connection. Finally, after introducing the application, this paper will discuss different design considerations, particularly related to privacy, security, and portability.

1.1 Overview of Identity

In this section, the concept of identity is introduced, and the different systems that are used to manage identities are discussed. The main focus will be on how this is achieved using digital technologies.

The identity of a person or place is the characteristics they have that distinguish them from others (9). In day-to-day life, people make various claims about their identity to other people. An example of this would be when purchasing an alcoholic beverage. To make the purchase, the customer claims to be at least 18 years old and, therefore, is allowed to make the purchase. To verify this claim, the person making the claim usually presents a credential such as a passport or some other identity card. These credentials are issued by trusted sources such as government bodies, and the person inspecting them can determine the validity by inspecting the document and ensuring that it comes from a recognised issuer and is valid.

This process can be carried out in much the same way in a digital environment. An individual can present a digital credential to prove that the claim they just made is true. This digital credential can be signed by the issuer, and anyone who checks it can verify authenticity by verifying the digital signature. This paper will focus on digital identity systems.

One key concept that is important for the remainder of this paper is that of an identifier. This term is often used interchangeably with identity; however, there is a difference. An identifier can be viewed as a reference to an identity, not the identity itself. For example, a Personal Public Services (PPS) number(10) in Ireland is a unique identifier that the Irish Department of Social Protection uses to identify people who live and work in Ireland.

The way identity works on the Internet is similar to this; most domains require users to create accounts, which function as identifiers, for each individual site. The result of this is that individuals must manage many different identifiers for every site they have previously accessed. The result of this model is that many people have more than 150 online accounts for which they must remember passwords.

An attempt to improve this system was the federated approach, in which an individual can access many domains with a single account. Microsoft single-sign-on is an example of this. The issue with this model is that it only works within the specified ecosystem and relies on a centralised institute or corporation such as Microsoft. This model is more common today, as many websites allow users to sign up using their Facebook or Google accounts. From a user-friendliness perspective, this is better, as only one account is needed, but it hands control over to the log-in service provider. So, if, for example, you lose access to your Facebook account, then you lose access to all the other services that are linked to that account.

A proposed solution to the problem of managing digital identity on the Internet is self-sovereign identity. A self-sovereign identity system would allow individuals to prove their identity on the Internet without relying on a third party such as Google and would allow them to only present the information that they wish to share to the other identities with which they wish to share it. Additionally, it should be interoperable and portable so that individuals can access any number of services using their own SSI.

This concept asserts that the user is fully in control of his own identity and is founded on ten principles laid out by Christopher Allen in his article "The Path to Self-Sovereign Identity"(3). These are:

No.	Principle	No.	Principle
1	Existence	6	Portability
2	Control	7	Interoperability
3	Access	8	Consent
4	Transparency	9	Minimalization
5	Persistence	10	Protection

Table 1.1: 10 Proposed Principles that SSI systems should follow

These principles can be used as a metric to compare the system described later in this paper and help illustrate the areas that require further work to develop. Some of the benefits of self-sovereign identity systems are as follows. The exchange of credentials is carried out in a secure peer-to-peer way so that only the identities involved in the exchange know what is being shared. Digital credentials are secure and tamper-proof through the use of cryptography. SSI credentials can be verified anywhere and at any time by anyone. Sensitive personal information is not stored centrally.

Self-sovereign identity has also become a popular topic in humanitarian settings. Almost 1 billion people do not have a formal identification(43). Digital self-sovereign identity systems can be used to issue identities to these individuals.

1.1.1 SSI Techonological Overview

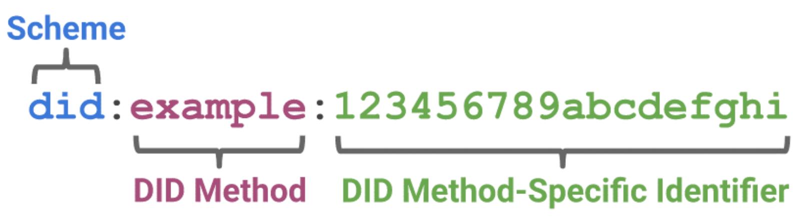
The concept of self-sovereign identity is relatively new and follows on from the invention of Bitcoin, which was the first blockchain-based cryptocurrency. (Blockchain will be explained in more detail in the State-of-the-Art Section later in this paper). Following the release of Bitcoin, many people interested in identity saw an opportunity to use this technology to solve the issues facing digital identity systems, as outlined above.

Much of this followed from work done as part of the Internet Identity Workshop (a conference for experts in internet identity), which held workshops on Blockchain Identity in its 20th meeting in 2015(29). The results of these discussions and the work that followed them resulted in the standards that are used as the building blocks of SSI today.

These building blocks are decentralised identifiers (DID), verifiable credentials (VC), digital wallets, and a verifiable data registry such as a blockchain. VCs are essentially digital credentials and work much the same way as real-world credentials. They consist mainly of an identifier, metadata, and claims and are signed by an issuer.

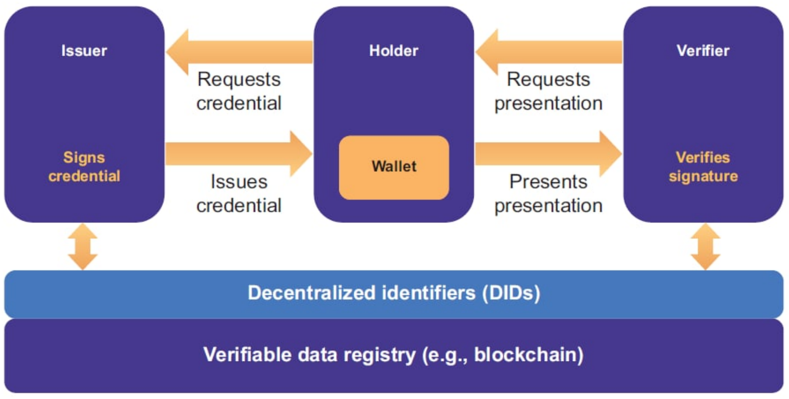
DIDs are a specific kind of digital identifier designed to be used on decentralised networks such as blockchains or peer-to-peer networks. The way in which they communicate with a given decentralised network is defined by the specific document DID method. This method defines how to create, read, update, and deactivate a DID document. DIDs are a W3C recommendation proposed on 31 August 2021, and a list of active development methods can be found on their website (41). Below is an example of a DID uniform resource locator (URL).

Figure 1.1: A basic example of a DID URL: Credit W3C(40)


Scheme
did:example:123456789abcdefghi
DID Method
DID Method-Specific Identifier

Such a URL will resolve to the associated DID document, which can contain arbitrary information, usually in the form of Javascript Object Notation (JSON) objects. Each DID has a public address on the associated data registry (e.g., blockchain) and a private key which is held in the holders' digital wallet. A public address is related to the private key through public/private-key cryptography. Only the holder of the DID should have access to the private key, as this is used to sign messages and prove that they are the owner of the DID. If the private key is compromised, then the DID is also compromised, so it should be kept secure by the owner.

Digital wallets serve two main functions for SSI. They hold the private keys which entities use to sign messages and prove that they own a DID and they are also used to store VCs which are associated with that DID. The diagram below illustrates how an individual can use DIDs and VCs to prove a claim on their identity.



The primary roles involved with exchange of verifiable credentials

Figure 1.2: There are three main actors involved in the exchange of VCs to prove a claim on identity. The above figure illustrates the process by which a claim is issued and verified using a verified data registry, DIDs, and a digital wallet: Credit Preukschat, Reed(35)

The diagram should be read from the centre out. The owner of a digital wallet requests a credential from the issuer. The issuer confirms that the person is who they claim to be based on their signature and checks that they qualify for the credential. Then sign the credential using the private key associated with their DID and issue the credential to the original DID holder who requests it. The individual then adds the credential to his digital wallet. Later, it is required that they present the credential.The DID holder then presents the credential to the verifier. The verifier can then confirm that the credential was signed by the recognised issuer by extracting the public key from the signature. If this is true and the credential metadata is valid, then the verifier can validate the claim made by the DID holder in the middle.

A real-life example of this would be obtaining a driving licence and presenting it to a police officer. The driver requests a licence from the issuing authority (e.g. National Driving Licence Service (NDLS) in Ireland). This authority ensures that the individual has passed their test and has not committed any offences which warrant a revocation of their licence and then issues the drivers licence to the individual. Then, when this individual is stopped by a police officer, they present the drivers licence, which the officer checks to see if it is valid and ensures that it is the correct person by comparing the image on the document to the person's face. This is usually carried out using a physical licence, but the process would be almost the same using digital documents. Except for digital documents, it is harder to fake a signature.

1.2 What is USSD?

As mentioned in the Introduction to this paper, an issue with existing SSI implementations is that there exists a technological barrier that requires a smartphone to access them, at least from the side of the verifier. The claimer could theoretically print their digital credentials on paper. Another issue with these systems is the offline validation of credentials, as the systems are inherently dependent on the Internet to function.

However, this is already an issue facing many companies in the developing world. For example, in subsaharan Africa, only 49% of the people are connected to mobile Internet services. This is increasing every year, but the price of smartphones and mobile data appears to be a barrier for many people(20). However, most people do have access to a standard feature phone that does not have internet connectivity, and more than 80% people have been shown to have access to a feature phone(37). This creates a problem for delivering digital products to populations in these parts of the world.

USSD is one of the solutions widely adopted in these developing economies to solve this particular problem. USSD is a GSM protocol that allows users to send text messages of up to 182 characters in a similar way to a Short Message Service (SMS). The main difference between the two protocols is that with USSD, a real-time connection is established between the sender and another device. This means that the sender can receive almost instantaneous responses from the service provider.

At a basic level the way this works for the user is that they dial a specific shortcode (e.g. *123*00001) which is given to them by the service provider on their phone. A two-way real-time connection is then created between the dialler and the service provider, and a menu is presented to the dialler. The dialler then enters a response by selecting which page in the menu they would like to view or by inputting information. A session is maintained with a unique session ID every time the shortcode is dialled and is automatically terminated if no response has been sent from either side within a 90-second window.

1.2.1 How USSD works

As mentioned above, the process starts when a user dials a specific USSD shortcode on their mobile device. This is an example of a man-machine interface or MMI code(27), a complete specification of MMI codes(8), but essentially they are specific codes that are used to access additional services on GSM. The specific USSD shortcode is registered with a mobile service provider and, when dialled, the user is connected to the specified service through a USSD gateway.

The USSD gateway resolves the shortcode, pulls the menu of the requested service, and then sends this menu back to the user. Any subsequent responses by the user are recognised as being part of the same session, so the user input is resolved in addition to the shortcode and sent to the application to handle.

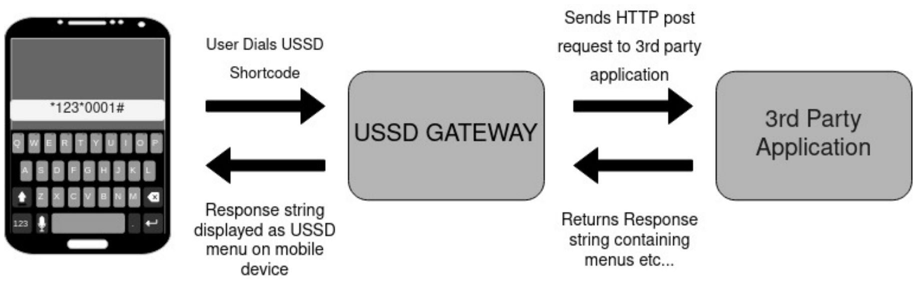

*123*0001#
User Dials USSD Shortcode
Response string displayed as USSD menu on mobile device
USSD GATEWAY
Sends HTTP post request to 3rd party application
Returns Response string containing menus etc...
3rd Party Application

Figure 1.3: Simplified diagram detailing the flow of calling a USSD application from a mobile phone

1.2.2 Why use USSD

The idea of using USSD as the interface for an identity application came primarily from personal experience of working in Malawi and seeing first-hand how comfortable people are using the technology and how important it is in people's day-to-day lives there. However, this first-hand experience is supported by data, as it is clear that USSD is still the main method by which people in developing regions such as Sub-Saharan Africa make digital payments and access financial services.

According to Statista, more than 548 million mobile money accounts were created in sub-Saharan Africa as of February 2022 (53). This number increased significantly during the Covid 19 pandemic(36). Many of these services operate exclusively through USSD applications (although some smartphone-based e-wallet applications are starting to be adopted by smartphone users). This shows that people are familiar with the technology and trust it enough to manage their finances through USSD mobile money applications.

Using USSD also effectively solves the issue of users requiring a smartphone to access on-line services and mitigates the issue of internet connectivity being required to verify decentralised identity claims.

Chapter 2

State of the Art

This chapter will explore in more detail the technologies mentioned in the previous chapter and provide examples. The areas that will be discussed are the digital identity, blockchain, SSI, and GSM communication protocols available on feature phones.

2.1 Digital Identity

This section will highlight some examples of well-developed digital identity infrastructures used around the world. These examples are not necessarily examples of decentralised digital identity or SSI but nonetheless are interesting to look at, as they illustrate the growth in trust OF digital identity systems around the world and also highlight the issue that this paper is trying to solve.

2.1.1 EU Covid-19 digital certificates

One of the most widely known and successfully adopted digital credentials introduced in recent years was the digital Covid-19 vaccine certificates used in the European Union (EU)(16). These are digital certificates that people can present to prove that they have been vaccinated against Covid 19, that they have had a negative test result or that they have recovered from the virus. Over the course of 2021, these certificates have been essential for accessing bars and restaurants and for travelling within the EU. The decision to use digital certificates was made to increase security and make it more difficult to alter certificates.

These certificates were created using the technologies mentioned above, namely VCs; however, they do not leverage a public blockchain or distributed ledger as a trust basis, meaning that they are not decentralised. Instead, the EU uses a standard public-key infrastructure that is verified through a gateway that was built specifically for this purpose.

This allows the EU to tightly control who holds the private keys that are allowed to sign certificates. It also reduces the complexity of the system, meaning that it could be rolled out faster.

When a person receives a certificate, it is signed by the relevant health authority in that country. The public key of the signer can be derived from this signature using cryptography. The certificate is then stored in a digital wallet on the user's smart phone with a QR code representing the certificate or on a piece of paper. When this QR code is scanned for verification, the certificate is sent to the EU central gateway for verification. This gateway holds the public keys of the authorities who are authorised to sign the certificates. The public key derived from the certificate signature is compared with the list of issuing authorities' public keys stored at the gateway. If the key that signed the certificate matches one of the keys stored in the gateway, then this means that the certificate came from a verified issuer and it can be validated.

The rapid uptake of this technology across the EU is exciting as it shows that the different governments in these countries are ready and willing to trust VCs for digital identity purposes and I hope to see the adoption of digital certificates for other purposes e.g. driving licences, passports, national id cards... However, the solution is not without flaws.

First, as stated previously, the implementation relies heavily on smartphones. This is not as much of an issue in the EU as smartphone penetration is very high, particularly among young people.

The other issue is that the system requires a large amount of trust in the EU. Fortunately, this trust exists among member states, and people living in European countries trusted the EU to implement this solution in a way that does not compromise their personal data. However, a decentralised identity is ideally able to work in a trustless environment where people may not have faith in their government to implement a system that does not compromise their personal data. This is why blockchain is often used as the trust basis for decentralised identity systems, as a verifier could choose who they trust to issue certificates themselves and verify certificates using blockchain as a trust basis.

2.1.2 Estonia E-ID

Estonia is probably the most developed country in the world when it comes to electronic or digital identification systems. Today, 99% of state services can be accessed digitally through the use of their electronic ID system; the only things that cannot be performed digitally are marriage, divorce, and real estate transactions. Furthermore, it allows users to access these services from abroad, meaning that people can vote, file tax reports, etc.

from anywhere in the world.

This system is also based on a public-key infrastructure where all citizens are issued with a mandatory national identity card, which includes a chip holding their keys. The chips on the card feature 384-bit ECC public-key encryption and require pin codes known only to the owner and a specific e-ID reader to use. This makes it very difficult to use another person's e-ID and ensures that no one can imitate someone else online without both their card and their pin codes.

All of the infrastructure behind the Estonian e-ID system is also open source and has been designed with interoperability in mind. This means that Estonian private sector companies can also use the e-ID system to authenticate users, meaning that people can log into online services with their national identity card or even use it as a loyalty card for brick and mortar stores.

National ID cards are issued to every citizen by government agencies, and they only receive their ID card and pin codes when they have sufficiently proven their identity to the issuing authority. These issuing authorities undergo regular audits to ensure that they maintain the highest levels of integrity and security.

The digital identity ecosystem in Estonia also implements a decentralised model. There is no single database that holds all the information about one person. Rather each organisation, e.g. banks, healthcare services, driving licence services, etc. have their own databases that interoperate through an open-source framework called X-road. Additionally, some of these records, for example healthcare records, are also recorded on the KSI blockchain to ensure that the records are transparent and cannot be altered.

This ecosystem is very impressive, and a full breakdown of how it works is beyond the scope of this article, but the systems in place in Estonia show the potential of digital identity in a developed country with enough investment in proper infrastructure.

However, as great as this system is, it still requires access to technology, such as a smartphone or laptop, to perform many actions with an e-ID. This system required significant government support and investment over time to implement. Investing in such advanced electronic identity systems is likely not a priority for many nations, and a more lightweight system would allow people in these nations to access at least some of the benefits provided by Estonian e-ID.

2.2 Blockchain

A blockchain is a peer-to-peer network that takes the form of a distributed ledger. It has a decentralised architecture that is hosted by a network of different nodes. These nodes host a full copy of the blockchain in question and a history of all transactions carried out.

Messages that take the form of transactions can be sent from one user to another; these transactions are represented as state transitions, and receipts are stored in blocks. When a user wishes to make a transaction, it is not carried out immediately. The transaction is propagated through the network using a gossip protocol, which essentially passes the transaction to a handful of nodes that verify the validity; these nodes then pass it on to other nodes until all nodes are aware of the proposed transaction. If the transaction is approved, it is then mined by special nodes called miners. This block is attached to the end of the chain of all previous blocks, resulting in the term blockchain.

Transactions are carried out using cryptocurrencies that are a form of digital currency that is specific to the blockchain for which they are designed. Each message/transaction costs a small amount of crypto to send. This fee is used to pay for the mining blocks. Mining can be quite a resource-intensive action, requiring electricity and expensive or specialised hardware. The cryptocurrency earned by miners can be seen as compensation or motivation for them to carry out this process.

One of the most important components of a blockchain is the consensus rules that govern what constitutes a transaction. A consensus algorithm then ensures that these rules are followed by each transaction recorded on the blockchain. The consensus algorithm must decentralise throughout the blockchain so that all participants follow the same rules. Understanding the specifics of how this algorithm works is not actually necessary in order to create applications which use a blockchain or to complete transactions, but it is good to know that they exist. After the network validates a transaction using the consensus algorithm, it is added to a pool of approved transactions that are waiting to be recorded in a new block. The transactions in this pool with the highest transaction cost are then mined first by mining nodes.

Not every blockchain user must run a node; in fact, most users do not. The way in which most users interact with the blockchain is through accounts, which are objects that hold cryptocurrency. Each account has a public address where anyone can look up and view the account balance and a history of transactions carried out by that account. The account is controlled with a private key that only the account owner should know. If the account owner wishes to create a transaction, they sign it using their private key and then send the transaction to the public address of the account with which they wish to interact.

The first commercially available and best-known blockchain is the blockchain that hosts the Bitcoin cryptocurrency and was introduced in 2008(28). Since then, the technology has become much more popular, and many different blockchains have been created for specific purposes. The blockchain that we will focus specifically on throughout this paper is the Ethereum blockchain(6).

2.2.1 Ethereum Blockchain

Ethereum is a general-purpose blockchain that was proposed by Vitalik Buterin in 2013. The primary difference between Ethereum and Bitcoin is that Ethereum was designed not only to be a distributed ledger to keep track of transactions, but also to be a general-purpose computing platform that is Turing complete(7). Turing completeness basically means that a computer or system can handle any computational problem, given that there is enough energy and memory available to it. This means that general-purpose computer programmes can run on the Ethereum blockchain; these programmes are specifically called smart contracts. Smart contracts are similar to accounts in that they have a public address and can theoretically send and receive cryptocurrency. They can also contain a generic key/value store that can be written to by anyone who has a key that allows them to do so.

Smart Contracts

Smart contracts are an important part of this project and a key component of the Ethereum blockchain. A simple way to think of smart contracts is to imagine that they are like classes in any other object-orientated programming language. Each contract contains attributes that are stored in its key/value store and methods that can be called by accounts or other smart contracts.

These smart contracts are written in high-level coding languages such as Solidity and are compiled into bytecode to be run by the EVM (Ethereum Virtual Machine). The EVM processes the state transitions that should occur as the result of a transaction, so, for example, how much Ether (Ethereum's cryptocurrency) each account present in the transaction should have at the end and what should be stored in the key value store.

What this means is that, in addition to being able to keep track of how much cryptocurrency exists and who owns it, Ethereum is also able to keep track of state transitions of general-purpose key value stores. This allows for a new type of transaction in which the value store of a specific smart contract is updated. These transactions contain at least the sender and recipient's addresses, the value to be sent, and a data payload.

The complexity of the operations performed by a smart contract has a direct impact on the transaction cost of calling one of its methods. Every operation performed by the EVM when executing a transaction has a predetermined cost in units of gas. If the account that calls the smart contract does not have enough ether in its account to cover the gas cost, then the transaction fails.

The ability to create smart contracts which execute business logic has led to the Ethereum blockchain's use as a platform to develop decentralised applications (DApps), which include Ethereum smart contracts as a back-end and a front-end web interface. The proposed decentralised identity solution discussed later in this paper is a DApp which uses an Ethereum smart contract as the user's DID and USSD for the front-end user interface.

ERC

Ethereum Requests for Comment (ERC) are documents written by blockchain developers to suggest new Ethereum standards so that other developers can write contracts that interoperate. Specifically, ERC refers to application-level standards, which could mean smart contract standards, name registries, URI schemes, library/package formats, and wallet formats(4). These ERCs are then reviewed by the Ethereum community in a process called the "Ethereum Improvement Proposal" (EIP)(13) to review the suggested standards and comment on them. The developer who proposed the ERC can then alter their standard in response to suggestions from the community. The most popular and well-known of these standards is probably the ERC 20 token standard.

ERC 20 details a standard for the creation of a new token(51). A token is a cryptocurrency that is not native to the Ethereum blockchain. The ERC20 standard provides methods to create tokens, transfer tokens, check the balance of an account, etc. It is considered a standard, meaning that any developer who wishes to implement their own token should use this standard.

ERC725/ERC734

This project uses a family of ERC standards, which is the ERC725 standard(52). This standard is defined as a general key value store and execution standard. The original version of this standard ERC725 v1 was specifically designed as a key manager and execution standard. The standard has since changed to become a more generic key / value store, and the original ERC725 v1 was renamed ERC734(50). ERC734 defines a set of methods to manage account permissions through the use of keys. The keys are derived from the public address of an Etherum account and are stored as a data structure by the smart contract.

```
struct Key {
     uint256[] purposes;
     uint256 keyType;
     bytes32 key;
 }
```

Listing 2.1: This snippet shows how a key data structure is defined for the ERC734 standard. The key struct contains a key item; this is a 32 bit hash of an Ethereum account's public key, a purpose item; which is used to determine what the associated account has permission to do, for example, a key of purpose 1 is a management key and can add other keys, and a key type item; this defines the schema used to define the key, e.g RSA ECDSA

The listing 2.1 above shows the attributes of an ERC734 key. ERC734 defines methods that allow one to add, remove, update, and read keys from a smart contract that implements the standard. When one of these methods is called, an event is emitted. This can be thought of as a console printing a log on a local machine, but instead of printing on a local machine, it is logged using the EVM. In addition to these methods to add remove read and update keys, there are execute and approve methods, which work in tandem.

This standard has since been replaced with another one, titled LSP-6, and was designed for the Lukso blockchain network, which was created by Fabian Vogelstellar and uses the Ethereum stack, but is a separate blockchain. The reason why this older standard is used for this project is how it interacts with the ERC735 claim manager standard, which is part of the same family of standards.

ERC735

The ERC735 standard is designed for claim management. ERC735 claims can be thought of as verifiable credentials, each has a unique claim ID and is linked to a specific smart contract. They are signed by the issuer, and this signature is used to verify the claim. The claims of ERC735 are specifically defined as a data structure with the following attributes.

```
struct Claim {
        uint256 topic;
        uint256 scheme;
        address issuer;
        bytes signature;
        bytes data;
        string uri;
    }
```

Listing 2.2: This snippet shows how a claim data structure is defined for the ERC735 standard. The topic attribute defines what the claim is in relation to; for example, claims of type 1 could be a name. The attribute of the scheme defines the scheme that should be used to verify the claim. The issuer attribute is the public address of the claim issuer. The signature is a message signed by the issuer, the signature must be a signed message of the following structure: $keccak256(abi.encode(identityHolderaddress, topic, data))$. Keccak256 is a hash function, and abi.encode encodes the values in bytes. The data attribute contains the signed encoded data. The URI attribute can be used to point to an off-chain address where the claim is stored.

Similarly to ERC734, this standard also includes methods for adding new claims to the smart contract, removing claims from a smart contract, and reading claims from a smart contract. With these methods, a user can manage claims about their identity, adding or removing them as they see fit.

By combining these two standards, it is possible to allow only those who have keys with purpose 1 to add or remove claims from an identity smart contract, thus giving the owner complete control.

The default ERC735 standard does not contain a method for validating claims. However, the way to do this is to verify the signature attribute of the claim. It is possible to retrieve the public key from a signed message(15), this recovered key can then be checked against the keys held by the claim issuer as defined by ERC734. If the claim issuer still has that key stored in their ERC734 smart contract, then the claim is valid.

These ethereum standards were chosen to implement the identity system described later in this paper for a number of reasons. Ethereum is one of, if not the most developed, blockchain infrastructures for creating decentralised applications, meaning that there are many resources available online and offline to help in the development process.

Another benefit of using Ethereum smart contracts is that all information is stored on the blockchain, which means that the user's phone does not have to store their DID.

However, the standards are out of date, and if this system were to be developed for use in the real world, it would be better to use one of the standards discussed below, as they are still receiving support.

2.3 Self Sovereign Identity

SSI is a relatively new field of study, and there is no single consensus on the methodology for implementing SSI. However, in general, the model shown in Figure **??** is valid, DIDs are used as a digital identifier and VCs are used to provide the proof of claims. A verifiable data registry is then used as the backbone through which parties interact.

In general, VCs and DIDs function in the same way for all implementations, and DIDs are unique and represent a single identity. VCs are used to prove a claim made about a DID. The main differences come from the verifiable data registry, which is the backbone that hosts the DIDs and records the interactions between them.

In their work on SSI Preukschat and Reed et al. identified four different types of verifiable data registry that could be used to host DIDs (35). These are...

- Adapting General Purpose Blockchains for SSI
- Using Specialised Blockchains Designed for SSI
- Using Standard Databases
- Using Peer-to-Peer methods

The way in which each of these systems functions as a verifiable data registry will be outlined in the following section by exploring some examples of each. The benefits and drawbacks of using each technology will then be discussed in the context of this project.

2.3.1 Adapting General Purpose Blockchains for SSI

The concept of self-sovereign identity came about as a response to blockchain technology and the possibilities it provided through its decentralised public-key infrastructure. As mentioned above, blockchains are interacted with through the use of accounts that each have addresses that are guaranteed to be unique through public-key cryptography. These addresses can therefore serve as a basis for DIDs, since they already satisfy the condition that each has a unique identifier and only the holder of the private key is able to control the account, thus enabling complete control of the identifier. These qualities make addresses on a public blockchain an attractive and relatively simple means of implementing an SSI system; however, there are issues with doing so.

One of these is that there is an inherent transaction cost to performing any action on most public blockchains, as outlined above. This is a serious issue for Ethereum, as transaction costs have become very high in recent times(57). If this cost is too high, then the identity system will lose its appeal, particularly in the context of this project, which aims to create an identity system for people who do not have access to smartphones, many of whom live in the poorest parts of the world.

Another potential issue is that all transactions recorded on the blockchain are visible and remain visible for all time. This poses privacy issues, as there is software available which is designed to link the pseudo-anonymous blockchain addresses (which are used as DIDs) to the real-world identities which control those addresses. Storing sensitive data on a blockchain is also problematic for this reason. This can be mitigated by storing the data "off-chain" and linking to it.

Key management is also a big issue when using a public blockchain as the basis for SSI. As mentioned above, the public blockchain address is managed through the use of a private key, and in the case of Ethereum, it is actually derived from that private key. This means that if an individual forgets his private key, then his digital identity is lost. Perhaps worse than this, if someone else learns this private key, they would then be able to pose as that identity. There exist methods to mitigate against losing private keys, such as using a seed phrase, but they do not fully solve this issue.

2.3.2 Using Specialised Blockchains Designed for SSI

As outlined above, general-purpose blockchains can be used to create SSI systems, but there are issues with using them specifically for this purpose. These blockchains have all the same qualities of a general purpose blockchain but are optimised to store and manage DID documents rather than just accounts which hold cryptocurrency. Evernym was founded in 2013 as probably the first special-purpose blockchain built for identity and remains the leader in decentralised identity(17). In 2016, the Sovrin Foundation was founded as a governance framework for SSI blockchain networks(39). Today, Sovrin contributes to the Hyperledger Indy codebase, which is a special purpose blockchain designed specifically for SSI as part of the Linux Foundation's hyperledger project(22). The Hyperledger Indy codebase was originally developed by Evernym before being donated to Sovrin, and the three organisations still have strong links.(19)

Hyperledgers are a type of distributed ledger, similar to a blockchain (in fact, sometimes they are referred to as enterprise blockchains). There are a number of differences between hyperledgers and general-purpose blockchains; however, the main differences are the following.

- Blockchains are public networks, which means that anyone can participate and view the entire transaction history of that blockchain. Hyperledgers are private networks, which means that only peers within specific organisations can participate, and only those with management credentials can see the transaction history

- Blockchains require a native cryptocurrency, and all transactions have a small cost in this cryptocurrency, which is used to reward miners for mining new blocks. Hyperledgers do not have a native cryptocurrency and there are no transaction costs. This is because the network is hosted by an organisation like Sovrin, who gets paid to operate it. This means that there is no need to incentivise mining through rewarding cryptocurrency

- Blockchains use a consensus algorithm to ensure that all transactions follow the consensus rules for that blockchain. Hyperledgers do not require a consensus algorithm, since all users are supposed to be within the same organisation, and therefore trustworthy.

The lack of transaction costs makes hyperledgers an attractive choice for SSI implementations, and the KIVA protocol, which is an "open source infrastructure for financial inclusion"(25), also used Hyperledger Indy for its pilot programme in Sierra Leone. This programme was designed to use blockchain to provide a financial history for people who do not have a bank account. This would help them build a credit score and apply for loans.

One issue with hyperledgers for SSI implementations is the fact that they are private networks, and only verified peers are allowed access. The overarching goal of this project is to improve access to digital identity, so using a technology that is not accessible to all did not seem sensible.

Recently, there have been other purpose-built identity blockchain projects that are built on top of more general-purpose blockchain technology, such as the KILT protocol(24), which is a blockchain built using the Polkadot(33) blockchain and specifically designed for managing identities. Polkadot is a blockchain infrastructure and ecosystem developed by Parity that allows developers to easily create parallel blockchains (parachains). KILT uses Polkadot because "it is the only technology to date that combines low fixed and definable network costs with aspects of the tamperproof, 'trustless' nature of blockchain" (24). This seems to be a best-of-both-worlds scenario, since any user can access this blockchain, and the low network costs make it attractive for use in an SSI application.

However, Kilt uses DIDs that are stored on the blockchain in a wallet. This could be an issue for this project, as client devices are assumed to be very limited in capabilities and are not guaranteed to be able to store DIDs.

Veres One is another similar project that has created an SSI-focused public blockchain that is optimised to keep costs low(49). The team behind the Veres One project was also involved in writing the W3C specifications for DIDs and VCs and has built the blockchain around these standards. The documentation states that the average cost to update a DID would only be around $ 0.25 on average, which is considerably less than the Ethereum network(57). The cost is also guaranteed to remain low, since peers who run special nodes called accelerators(48) are paid to create and update DIDs on behalf of other peers. The rates they charge are determined by the blockchain board of governance. Additionally, fees are waived for humanitarian purposes, such as issuing digital credentials to refugees or improving access to services for the unbanked or underbanked, such as the people that the KIVA project aims to help. However, one potential issue with this blockchain is speed, as the documentation claims that it can take a few minutes to create an identifier. This is not too much of an issue, but it is definitely slower than deploying a smart contract-based identity on Ethereum.

Unfortunately, the development work on the proof-of-concept for this project had already begun before learning about these two blockchains, so they were not considered for development purposes.

Purpose-built blockchains seem to offer all the benefits of a general purpose blockchain but do not necessarily have all the associated drawbacks, making them an attractive option for developing SSI applications.

2.3.3 Using Standard Databases

Using a standard database is also an option, and the Estonian system described above uses standard databases as a basis. However, using a centralised database removes the benefits of a decentralised system by introducing a single point of failure and means that the system is more vulnerable to attack. It also means that if the company or organisation that is maintaining that database closes and stops maintaining it, people's data is lost.

In Estonia, they mitigate this issue by storing information in a number of centralised databases and have created a data exchange layer called x-road to allow these different databases to communicate(12). This means that there is not only a single point of failure, and the overall system works even if one of the databases is down. This works in Estonia because they have systems in place to audit each of the database providers to ensure that there are no weak links, and the system has full government backing.

However, Ultimately, with centralised databases the data stored are under the control of the organisation that hosts the database. In recent years, there have been many concerns raised over the amount of personal data held by large corporations, and there is a lack of trust for these companies. Regardless of who is in charge, if data are stored in a centralised database, the owner of that database has a lot of control over user data, which is against the governing principles of SSI(3).

2.3.4 Using Peer to Peer Methods

Some new technologies have emerged that do not require a distributed ledger or blockchain as a data registry for DID documents. These DIDs and DID documents can be shared directly peer-to-peer, and authentication is carried out by the DID documents. In such a model, the DID registry is actually the DID wallet as the wallets of the peers who are exchanging credentials.

This approach shows great promise, as it is highly scalable, fully decentralised, and does not require a blockchain, meaning that there is no transaction cost to share credentials. An example of a comprehensive architecture for peer-to-peer DID is the KERI protocol developed by Dr. Samuel Smith(23)(38). This protocol also relies on the overarching concept of public and private keys but uses the concept of self-certifying identifiers to prove ownership of an identifier.

Self-certifying identifiers are a type of DID that does not require a blockchain or any other form of distributed ledger to prove ownership. Instead, the identifier is proven to belong to a public key through only cryptographic algorithms. The controller (e.g. wallet holder) issues their own Identifier by generating a public/private key pair and cryptographically binding it to an identifier. The public key is then used as the prefix of the identifier (see Figure 1.1, where the prefix would be found at the beginning of the identifier specific to the DID method).

This self-certifying identifier is then used as the "root of trust", meaning that if someone has the private key to control an identifier, you can trust that they own it. When an identifier is created and bound to a key pair, this is called an inception event and is the first event in a chain. This chain of events is called a Key Event Log (KEL). Other events are then appended to this log such as rotation events where the controller rotates control of an identifier from one key pair to another, and transaction events when the identifier interacts with another identifier; an example of a transaction event might be requesting a VC from another identifier.

This KEL is stored within the controller digital wallet and not in a distributed ledger or a centralised database of another kind. When two identifiers interact, they share a key event receipt log (KERL) with each other. This is a copy of the KEL except that only digests or receipts of each transaction are included, and not all of the relevant data. To check if a credential shared that was valid, the verifier can request a KERL from the issuer, and they should have a matching transaction which shows that they issued the credential. Or if the credential was revoked, they should also have a recorded event showing that.

One issue with such a system is duplicity, which means sharing contradictory information with different peers. Since there is no distributed ledger backing the system, you cannot know whether the KERL that has been sent to you has been altered or modified in any way. To mitigate against this, KERI proposes the use of witnesses in the peer-to-peer network; these witnesses store the KERL of different identifiers with which they have interacted. So, say, for example, Kate wants to send a different KERL to two different certificate validators, Eve and Joe. Eve and Joe can then send the logs which they have received to Doug, who is designated as a witness; Doug will then see that the logs sent to Eve and Joe are different and flag that Kate is being duplicitous. This is visually shown in the figure below 2.1

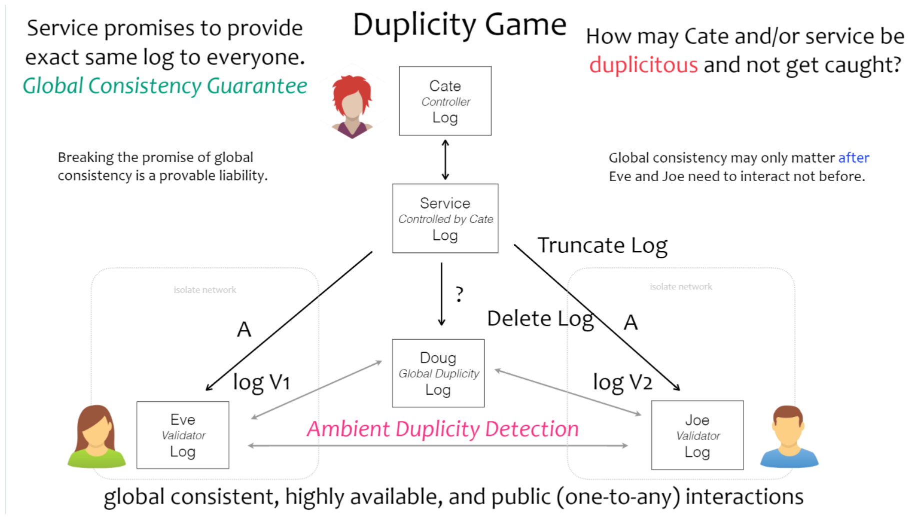

Service promises to provide exact same log to everyone.
Global Consistency Guarantee
Duplicity Game
How may Cate and/or service be duplicitous and not get caught?
Cate Controller Log
Breaking the promise of global consistency is a provable liability.
Global consistency may only matter after Eve and Joe need to interact not before.
Service Controlled by Cate Log
Truncate Log
isolate network
isolate network
?
A
Delete Log
A
Doug Global Duplicity Log
log V1
log V2
Eve Validator Log
Ambient Duplicity Detection
Joe Validator Log
global consistent, highly available, and public (one-to-any) interactions

Figure 2.1: Diagram showing how a global witness (Doug) can be used to detect if an agent (Cate) is being duplicitous when interacting with two or more different verifiers (Eve and Joe)

Peer-to-peer implementations such as the KERI protocol show a lot of promise and are probably the closest thing to true self-sovereign identity implemented to date. The problem with using this sort of technology in this project is that it relies on the mobile wallet stored in each edge device, e.g. a smartphone, to store not only credentials but also the KEL of each identifier. This makes it more difficult to implement a system using these methods for non-smart edge devices, such as "dumb" phones, which is the focus of this project.

2.4 GSM Protocols

In this section, we will discuss the possible ways users can access a digital identity stored online using GSM protocols. That is, they do not require a smart device with Internet connectivity. The three methods that will be discussed are Short Messaging Service (SMS), USSD, and Interactive Voice Response (IVR).

SMS

SMS is a messaging service that exists on most mobile phones and allows peers to send short text messages (under 160 characters) to each other. It is a GSM protocol which was first formalised in 1985 and is one of the most popular and well-known ways of sending text messages. To send a message to another peer, the sender of the message must first know their phone number. The message is then sent over the sender network provider's network to a service centre. Messages are temporarily held in these service centres and then forwarded to the recipient.

SMS can be used to connect users to online services using an SMS api such as Twilio(46). This works by creating an account and registering a phone number with the api provider. When this phone number then receives an SMS, the api calls a webhook, which is usually a developer-defined http request. This allows users to interact with basic web applications by sending messages to the registered phone number.

This strategy would work for the use case defined in this paper; however, it creates some issues for the user. The first of these is that every time the user sends an SMS there is a charge, which is defined by the network provider. This is usually quite small (18c per message with Three Ireland)(44) but if multiple messages are required to perform a task, the cost will add up.

SMS is also only a one-way connection. This means that the sender must wait for a response. Again, if a number of messages are required to complete a session, this could take a while and is not great from a user experience point of view.

The one-way connection also presents another issue for users. The webhook which connects the user to the http endpoint will likely use string processing to handle the users request, since all they can send is a string. This means that the user must memorise the format for messages, and if they make a mistake, their request will likely fail. Natural language processing methods could potentially be used to mitigate against these errors, but would add a lot of complexity to the application and further increase latency and thus response time.

Furthermore, SMS messages are often not encrypted when they pass through the network or the service centre. This has made SMS service centres a target for hackers in the past(11) as they contain a large amount of unprotected data that could be valuable to hackers.

SMS was discounted as an option to handle responses in this project for these reasons; however, it still has its merits and could be used instead of push notifications.

IVR

Interactive voice response services are based on the ability of a telecom network to transmit audio as a phone call. The user dials the number they wish to reach and is met with an automated voice response. The IVR service then presents the dialler with a number of options. The dialler then selects the appropriate response by using the keypad on his mobile phone or by speaking. This is done using dual-tone multifrequency signaling (DTMF)(1) or speech recognition.

This type of automated answering service is used by many businesses around the world to improve call centre efficiency and automate customer responses.

DTMF works by using the keypad of the user's phone. The buttons are segmented, and when a button is dialled, a sound is played with a specific frequency. This sound is transmitted over the telephone network and picked up by the receiver. The audio signal received is then decoded by the receiver to determine which key was pressed by the dialler.

The user response can then be passed as an http request in the same way as with the SMS example above. Twilio also provides a voice API, which allows developers to build an IVR service to handle user requests. Using this kind of system solves the waiting-time issue and restricts the user's input, so it is more difficult for them to enter a command which is not valid. There is also less security concern compared to SMS.

Security concerns are similar to those when transmitting voice data across a GSM network, as the content of the conversation is not stored in a service centre. Voice data is also usually encrypted while travelling over GSM networks, but with relatively weak encryption A5 / 1 or A5 / 2. However, one issue with using an IVR system is scalability. The voice response must be re-recorded in the local language for every region that the service is deployed. This requires more effort than simply translating text, which can be done by a computer programme if needed.

USSD

USSD is explained in the Introduction section 1.2. In terms of utility, it sits in between IVR and SMS as there is a two-way connection allowing instantaneous responses to be sent back to the dialler, but it requires users to input their response by inputting a text string.

However, one benefit of using this technology over SMS is that it does not cost the dialler anything to use USSD. Instead, the entity that wishes to provide a service pays the telecom provider, who in turn connects users to their service through their USSD gateway.

It is also much easier to provide a USSD service in multiple languages compared to an IVR system. This is because the only thing to translate are a few short text menus. This could be configured very easily just by using a JSON file that has the responses for each language defined and by providing a different short code to dial for each language. The short code can be checked when the server receives a request and the correct menu can be selected and sent to the user.

In terms of security, USSD is somewhere between IVR and SMS. Data are encrypted while travelling across the network with USSD; however, encryption is usually quite light and easily broken on GSM networks, for example A5/1 and A5/2. However, this can be improved by programming the sim card as suggested by the International Telecommunication Union(42). USSD is also a familiar technology that many people trust to make financial transactions every day, so it makes sense to use a trusted service.

USSD was chosen to implement this project due to its scalability and because it is already trusted for secure processes such as banking in many countries.

2.5 Related Projects

The previous sections have given an overview of the technologies that will be used in the implementation of this project and explained the reasoning behind these choices. In the following paragraphs, we will explore two projects that make use of these technologies to implement solutions related to this project and compare their similarities and differences with this project.

2.5.1 Kiva Protocol

Kiva is a non-profit organisation that was founded in 2005 with the aim of increasing financial inclusion through open source digital infrastructure. The organisation started by issuing microloans to people in Kenya, but has since changed focus and in 2019 launched the Kiva protocol in Sierra Leone. The protocol aims to provide Sierra Leoneans with a means of developing a credit history and was launched with the support of the Sierra Leone government.

The underlying infrastructure of the Kiva protocol uses a permissioned blockchain built with Hyperledger Indy. This means that Kiva and/or the Sierra Leone government can specify which peers are allowed to issue credentials, and other peers do not have permission to do this. DIDs are issued to people who have a government-issued national ID card, and these are linked to a public/private key pair. This is done through off-line channels, such as at polling stations or when the national ID card is issued. The national ID card is the basis for the DID, but other information such as biometrics is attached to it.

If the person registering has a smartphone, the key pair and associated DID are saved in a wallet application on their phone. If the person does not have a smartphone, the keys are stored in an encrypted database under the guardianship of Kiva and the government. People can remove these keys from the database at any time they wish. Kiva also stores backup keys that the person can access if necessary.

The lenders are then required to report transactions using the protocol. These loans are recorded as sub-DIDs, which are linked to both the lender and the person receiving the loan. Then, if someone wants to access the credit history of a person, they request permission from that person. They can then give or deny this permission to the interested party. If access is granted, then the interested party will then be able to view the sub-DIDs associated with the person until the permission is revoked.

This is probably the most developed example of a digital identity system deployed in a developing economy. The identity model used sits somewhere between federated identity and SSI since a government issued ID card is still required and only approved entities can issue VCs, but the underlying technology gives users a lot of control over who can view certain aspects of their identity. A lot of data is stored by Kiva; however, this is mainly due to the lack of penetration of smartphones in Sierra Leone, and this is an issue faced when developing a prototype for this project.

This system is a good example of what can be achieved using open source standards in an environment where there is cooperation from the government and trust for the government among the population. However, the system would not be applicable in a scenario where the government is resistant to digital identity or does not have the trust of its population.

2.5.2 Leaf Global Fintech

Leaf is a project currently supported by the Unicef Innovation Fund(?), which invests in companies that have the potential to improve the lives of children. It aims to improve financial inclusion for people who do not have a bank account through the use of blockchain and cryptocurrencies.The system is aimed at refugees and people who do not have a bank account but may still need to send or receive money internationally. Leaf allows users to send and receive money in the form of cryptocurrency through a digital wallet, which they manage using either a smartphone app or a USSD interface.

The system is backed by the stellar blockchain, which is specifically designed for cross-border money transfers. It uses stablecoins, which are tokens that have their value tied to a real-world currency. This means that when Leaf customers in Rwanda use the service, it is familiar, as it appears that they are using their national currency.

Unfortunately, it is not clear how Leaf is managing the digital wallets of users who access the service through USSD using a feature phone; only PIN authentication is used and the service is fully hosted on Amazon Web Services.

The project has since grown to be able to provide blockchain-based loans to customers and has supported more than 6,800 users as of December 30, 2021.(18) The company was acquired by IDT Corporation, which is a large multinational. This success shows that the use of USSD to provide users with access to the blockchain is viable and secure enough for financial transactions.

2.5.3 World Food Programme (WFP) Building Blocks

The World Food Programme (WFP)(56) is "the leading humanitarian organisation saving lives and changing lives, providing food assistance in emergencies and working with communities to improve nutrition and build resilience". It is a branch of the United Nations and supports projects in 84 countries.

In 2017, WFP launched a proof-of-concept programme called building blocks with 100 people in a village in Pakistan(55)(54). For the proof of concept 100 beneficiary accounts were created on the Ethereum blockchain and each account was transferred tokens representing money or food items. Participants received a random identifier between 1 and 100 that was mapped directly to their Ethereum account.

When they entered a participating merchant's store, beneficiaries provided their random identifier to the cashier, who then entered this number into a web application. The beneficiary then received a one-time passcode (OTP) via SMS from WFP building blocks. The beneficiary shares the OTP with the cashier, who then enters it into the web app and sends it to building blocks. If the OTP is valid, building blocks check the beneficiaries account to see if they have enough tokens for that transaction, and if so, record a transaction on the blockchain using the beneficiaries private key and send confirmation to the merchant over the web app.

The transaction history recorded on the blockchain was then used to determine how much money each merchant was owed, and building blocks then paid them directly.

After the proof of concept phase the application was scaled up to use in a refugee camp in Jordan. Building Blocks switched from the public Ethereum network to a private permissioned Ethereum blockchain provided by Parity. This permissioned blockchain had a higher transaction throughput and no transaction charges.

The webapp used by merchants was replaced by iris scanners which the UN had already deployed in the refugee camp. Refugees only needed to scan their iris when collecting food or supplies and building blocks would record the transaction on the private blockchain allowing the refugee to recieve their food.

In both iterations of the system building blocks held the private keys for all beneficiaries in custody and performed transactions on the blockchain on behalf of the beneficiaries.

2.5.4 Summary

These three projects show that technologies such as blockchain can be used to benefit people's lives in developing economies, and Leaf in particlar shows that it can be successfully paired with older technology such as USSD, which is more prevalent in these developing economies. The following Table 2.1 is a comparison of the three projects and highlights their approaches to some of the key aspects of this project.

	Kiva	**Leaf**	**Building Blocks**
WalletAccess	Users can access their DID either through a smartphone wallet or through guardianship model	Users can access currency wallet through either USSD or Smartphone App	Beneficiaries do not directly access wallet themselves
Purpose	Credit history tracking for citizens of Sierra Leone	Cross border payments with low transaction charges for refugees and unbanked in East Africa	Tracking distribution of food aid in refugee situations
Accessibility	Anyone in Sierra Leone with a government issued national ID can avail of Kiva	Available to users in Kenya, Rwanda and Uganda, only an image of the customer is required to sign up	Refugees registered by the UN using biometrics
Identity	Federated and linked to government ID. Utilizes hyperledger indy for signing claims using DIDs	ID is not the focus, used for money transfer	Beneficiaries linked to Ethereum account, used to track transactions

Table 2.1: A comparison of the two projects discussed, comparing how each project addresses Wallet Access, the purpose of the project, who can access the service, and what identity model is used.

Chapter 3

Design

3.1 Overview of the Approach

As explained in the previous chapter, the identity management system that will be described in the rest of this report is implemented using Ethereum smart contracts following the ERC734 and ERC735 standards and uses USSD to provide a user interface that can be accessed using any feature phone.

There are several design considerations and constraints to consider when implementing such a system. For this project, the focus was on creating a proof-of-concept to show that it is possible to create and manage a smart contract-based identity system using USSD. For this reason, the application was kept as simple as possible.

The application was broken down into four different components before starting to build the prototype. These areas were cloud services, the USSD API, smart identity contracts, and the application that connects everything else together. Additionally, there needed to be two different user interfaces, one for verifiers and one for normal users.

The user interface needed to allow users to create a new identity smart contract and to add, remove, or present claims about that identity. Verifiers must be able to verify the claims presented by users. This interface is presented to the user or the verifier through USSD menus.

Since the system is designed for use with feature phones, all user interactions must be handled on a cloud platform. This includes authentication, data sharing, and data storage. Additionally, the application must be hosted online so that the USSD API can interact with its http endpoint.

The remainder of this chapter will focus on the design choices that were made when designing a prototype identity system for this project and why those choices were made. An overview of the application design is shown in Figure 3.1 below.

3.1.1 Cloud Services

The cloud component for this application must perform three main tasks. These are hosting the JavaScript application, a database, and user authentication. The database is needed to store information on users, verifiers, credential issuers, and sessions.

Storing session data collection is necessary to keep track of which pages should be presented to each user and whether they have been authenticated. It stores a unique session ID for each session and the page number of the last menu presented to each user. This, in combination with the user's input, is used to determine which menu to present to the user or which smart contract method to call.

Since the users' edge devices are assumed to be unable to securely store the public/private key pair for their Ethereum account, this must be stored in the database. Additionally, it is difficult to enter long or complex strings using a feature phone, meaning that it is impractical for users to enter claim information, such as the signature manually, and may lead to incorrect information being entered. For this reason, claims which are verified by the issuers should be stored on the database until they are added to the user's smart contract.

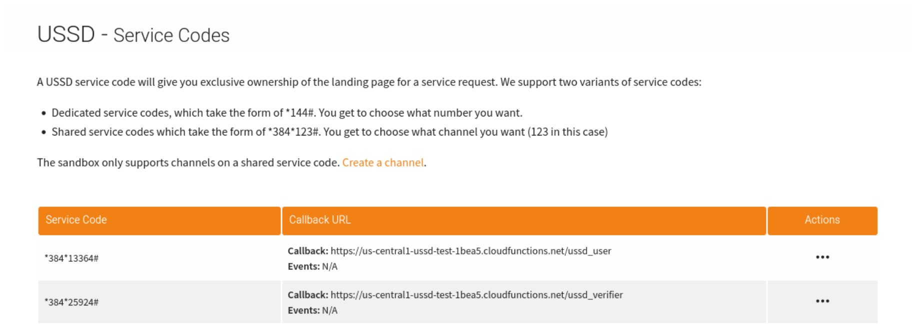
All of this can be done with a database and an application hosted in the cloud. There are a number of cloud service providers that can offer this functionality, but for this project, Google's Firestore suite was chosen to perform these tasks. Firebase Cloud Firestore was chosen for the database and the application is hosted as two Firebase Cloud Functions(21).

Cloud Firestore is a no-sql database provided by Google. Data are stored in documents, so, for example, each user would have one document related to them, and within that document there are a number of data fields describing the user. Documents are organised into collections that can be used to group similar documents and organise the database. Cloud Functions are " a serverless framework that lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests". It allows developers to store their JavaScript or TypeScript code in the Google cloud. This code is then executed in response to a listener, so in this case a HTTP POST is sent to the URL which the function is listening on. One benefit of using the Firebase suite of tools is that there is a local emulator suite included with the Firebase. This speeds up prototyping as there is no need to wait for the functions to be deployed online.

3.1.2 USSD Provider

Africa's Talking(2) was chosen as the USSD api provider to build the prototype application. Africa's Tallking is an API provider with the goal of providing infrastructure to support the developer community in Africa in building connected applications. They offer a number of communication APIs for a range of communication technologies such as USSD, SMS voice, and so on. and provide a simulator app which allows developers to test the applications they have built and see what they would look like once deployed. To use the API, the developer must first create an application and host it on a http or https url. This URL is registered on their platform and linked to a USSD short code. When the shortcode is called using the simulator app, the contents of the text response and the phone number are resolved into a http POST request to the provided URL. Africa's Talking was chosen for two reasons; it is a very popular provider, so there are many available resources online; it offers a simulator web interface that allows developers to test their application's user interface without setting up a gateway service.

Using one of the mobile money provider APIs (e.g. Airtel) was also considered, as these are assumed to be more secure; however, it was not possible to sign up to use their API or view the documentation with an Irish phone number, as one from one of their service areas was required. Designing the system using one of these APIs might also have had the negative effect of locking access to the service to users using that telecom provider's SIM card. These APIs also appeared to focus on businesses and provide them with the tools to interact with customers using mobile money services. Prototyping with Africa's talking is quite simple, all the developer must do is enter a callback URL that will receive a POST request from the API as shown in Figure 3.2 below. They can then test the application they have built by using the simulator and dialling the code attached to their callback URL as shown in figure 3.3 below.



USSD - Service Codes

A USSD service code will give you exclusive ownership of the landing page for a service request. We support two variants of service codes:

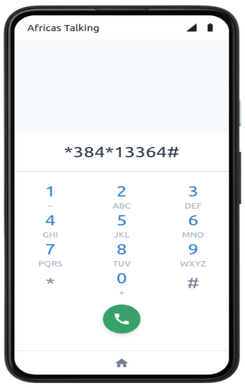
- Dedicated service codes, which take the form of *144#. You get to choose what number you want.
- Shared service codes which take the form of *384*123#. You get to choose what channel you want (123 in this case)

The sandbox only supports channels on a shared service code. Create a channel.

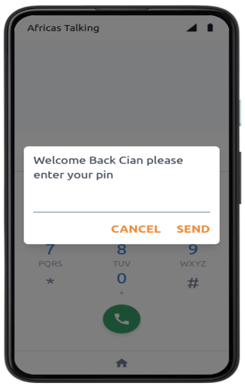
Service Code	Callback URL	Actions
*384*13364#	**Callback:** https://us-central1-ussd-test-1bea5.cloudfunctions.net/ussd_user **Events:** N/A	•••
*384*25924#	**Callback:** https://us-central1-ussd-test-1bea5.cloudfunctions.net/ussd_verifier **Events:** N/A	•••

Figure 3.2: Screen for registering callback URLs as USSD shortcodes on Africa's Talking

Figure 3.3: Africa's Talking simulator interface



(a) Dialing short code on Afria's Talking Simulator



(b) Result of dialing short code on simulator.

3.1.3 Main Application

The backend application for this project performs three main tasks. It receives commands in the form of strings from the USSD API, parses those commands, reads from, or writes to the database, and calls smart contract methods. These tasks can be accomplished using a variety of programming languages; however, Firebase cloud functions must be written in JavaScript or TypeScript. This application did not require the extra features offered by TypeScript, so the application was developed using JavaScript with a Node.js(32) runtime environment.

To perform the required tasks, a number of external packages and libraries were required. Express.js(31) was used to handle HTTP post requests sent by the USSD API and to send the responses back to it. ”is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.”

The Firebase Admin SDK was used to manage the connection to Firebase Cloud Functions and the Cloud Firestore. This is a set of server libraries that allow server-side applications to interact with Firebase services with administrator privileges. The Admin SDK is required because the application runs exclusively from cloud functions, and this is the officially recommended way of accessing other Firebase services through cloud functions.

The Web3.js library (14) is used to connect the application to the blockchain and interact with smart contracts. This is probably the most popular library for interacting with Ethereum-based blockchains and is maintained by the Ethereum Foundation. The main requirements of this app were to create and encrypt an account for each user, to deploy and call smart contracts, to sign messages using these accounts, and to encode the data that are passed to the smart contracts. Web3.js provides functions to accomplish all these tasks, in addition to excellent documentation and a large community of users, making it the obvious choice for this project.

The Javascript Crypto library was also used to securely encrypt or hash sensitive data before storing them in the firestore database.

3.1.4 Smart Contracts

Ethereum smart contracts can be written in several programming languages, but Solidity and Vyper are the two most popular. Both are high-level programming languages and are compiled into bytecode to run on the Ethereum virtual machine. Solidity is an object-orientated language, similar to C++ syntax, and supports features such as inheritance and using other smart contracts as libraries. Vyper has fewer features than Solidity and is more similar to Python in its syntax. It does not support inheritance.

Solidity smart contracts were used because they needed to inherit from the base ERC734 and ERC735 smart contracts. Previous implementations of these standards already exist, so it did not make sense to rewrite smart contracts from the ground up. Instead, the already implemented OnchainID(30) identity and claim issuer contracts(5) were used. This was decided to save development time and prevent unnecessary work.

The smart contracts were tested to ensure that they functioned as desired before integrating them into the application. Contract testing was performed using the Remix integrated development environment (IDE). This is a browser-based IDE specifically designed for deploying and testing Ethereum smart contracts, and allows developers to call contract methods and see the results in browser just by filling in the parameters in a drop-down menu.

3.1.5 Ethereum test network

As mentioned in previous sections, performing any transaction on the Ethereum main blockchain has an associated cost. Additionally, smart contracts remain deployed at a given address forever, meaning that if there is an exploitable weakness in the smart contract, it may be vulnerable to attack and cannot be removed to fix the error. Therefore, to avoid spending real money while developing DApps, there are a number of Ethereum test networks. These test networks function in the same way as the main net, but use a free version of the native cryptocurrency, allowing developers to deploy and test their smart contracts without spending real money.

There are several test networks that aim to simulate the main Ethereum network, but for this project, I decided to use the L14 test network, which is a test network for the Lukso blockchain(26). Lukso is a blockchain that was invented by Fabian Vogelstellar, the creator of the ERC725 standard. It uses its own native cryptocurrency LXE but runs on the Ethereum stack, which means that smart contracts work the same way as they would on the main Ethereum network.

This test network was chosen for a couple of reasons. The first of these is that the network was designed as a digital lifestyle blockchain with the ERC725v2 smart contract at the core, so it seemed fitting to develop using this blockchain. Another reason is ease of use. To obtain the test cryptocurrency for this network, all one needs to do is paste the public key of the account to fund into a faucet page. 5 LXE tokens are transferred to that account, which is more than enough to test this application.

3.1.6 Local Testing Setup

The above tools were used to test the application when it was nearing completion, but deploying the application online every time a change is made is very time consuming and significantly slows development. For that reason, the following tools were used to prototype and test locally on my desktop. Ganache(45) was used to test the functions that interact with smart contracts. Ganache is a programme that allows users to simulate an Ethereum blockchain locally on their computer. Automatically creates ten accounts, each with 100 fake ether to use to test smart contracts locally. As mentioned above, the Firebase Software Development Kit (SDK) includes a local emulator suite, which creates a local instance of the Firestore database in memory and hosts the cloud functions at a port on the local host. When testing the programme locally, it was not possible to use the Africa's Talking API as it required an online callback URL. Instead, Postman was used. Postman is a tool for testing web APIs and can be used to send http requests to APIs or, in this case, programmes and read the response(34). The Africa's Talking URL resolves the phone users' input into a http POST request, so it was possible to test the application by directly sending it a POST request with the body replicating what the API would produce.

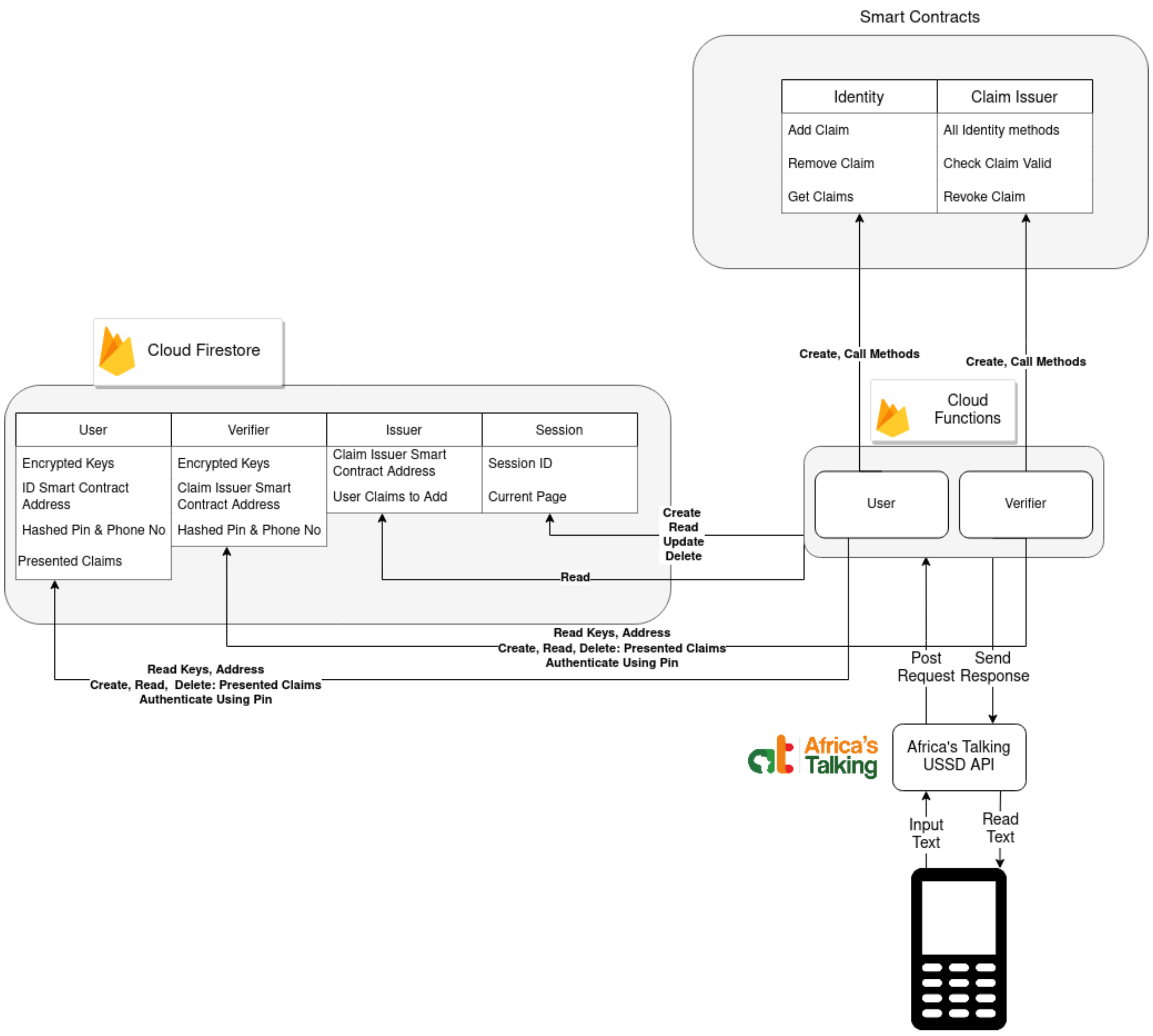


Figure 3.1: Diagram of Application Design: Arrows represent actions. Headings in Cloud Functions Bubble represent the two hosted functions. Headings in Cloud Firestore represent Collections, subheadings represent types of data stored. Headings in Smart Contracts represent contract name, subheadings represent contract methods

Chapter 4

Implementation

In the previous chapter, the tools used to test and prototype this application were introduced. In this chapter, the specifics of how different elements work and interact with each other will be explained in greater detail.

4.0.1 USSD Api

Unfortunately there is not a lot of documentation explaining the specifics of how the Africa's Talking USSD API works behind the scenes. However, there is some information in the developer documentation that can be interpreted. As mentioned in previous chapters, USSD operates on a session basis, and a real-time connection is made between the client and the server for the duration of this session. The Africa's Talking API generates a unique ID for each of these sessions and sends the server http POST requests with the following parameters.

- sessionID (string): This is the unique session ID generated by the API and is used for the duration of the session until terminated by the server or client
- phoneNumber (string): This is the phone number of the client who dialled the USSD code
- serviceCode (string): This is the USSD short code entered by the client
- networkCode (string): The telecom provider of the phone number that dialed the USSD code
- text: The text input from the client device.

A session is initialised by the user dialling the service code, and the API generates a session ID and sends a POST request to the server with an empty text field. This session remains open until one of three things happens. The API receives a message starting with the keyword "END" from the server, indicating to terminate the connection; the API receives an error message from the server; the session times out.

Each input from the user is recorded during the session, and an asterisk is appended to the end of each input to separate it from the previous.

This is all the information needed to be able to use the API in this application.

4.0.2 Cloud Services

The cloud services can be broken down into two areas, application hosting and database. The hosting element is rather simple as there are two Firebase cloud functions configured that receive http POST requests from the USSD API.

The database used is Firebase Cloud Firestore, which is a no-SQL database that stores information as JSON objects. Data are recorded in documents which are grouped with similar documents inside collections. Documents can have an arbitrary number of attributes and can also contain sub-collections of documents.

The layout of the data in the cloud firestore for this project can be seen in Figure 4.1.

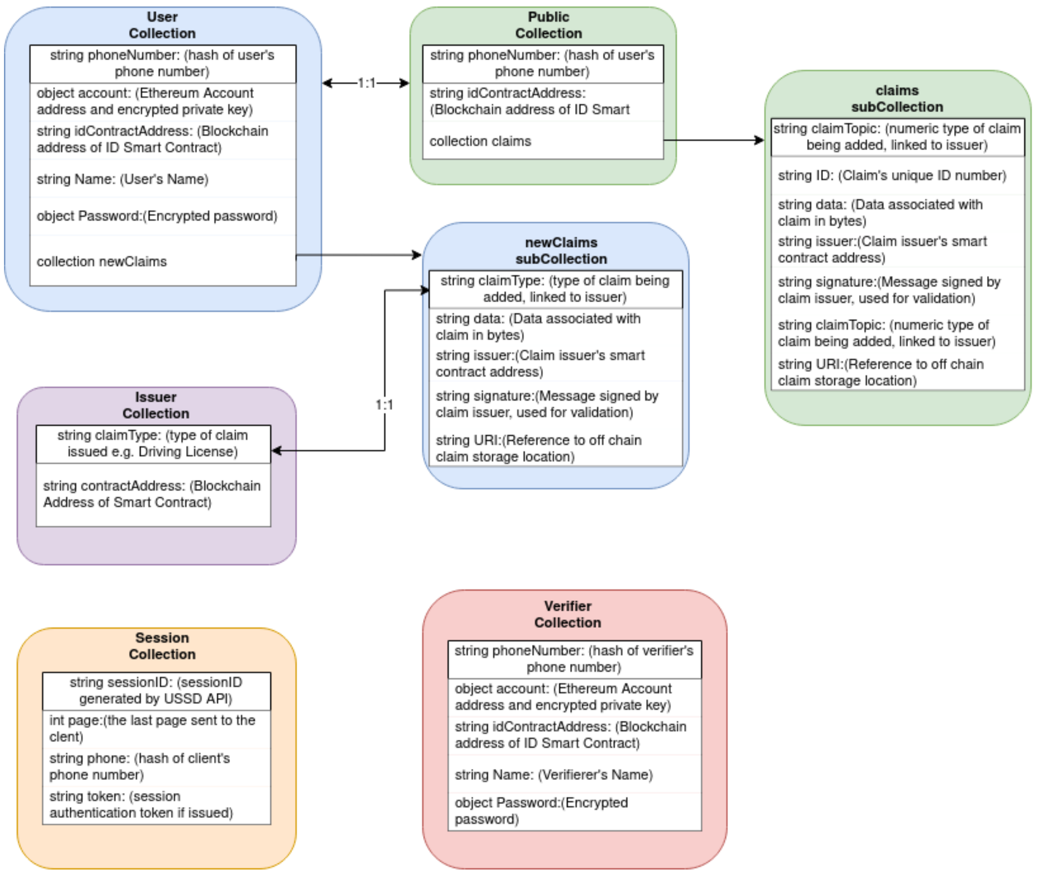

User Collection
string phoneNumber: (hash of user's phone number)
object account: (Ethereum Account address and encrypted private key)
string idContractAddress: (Blockchain address of ID Smart Contract)
string Name: (User's Name)
object Password:(Encrypted password)
collection newClaims
1:1
Public Collection
string phoneNumber: (hash of user's phone number)
string idContractAddress: (Blockchain address of ID Smart
collection claims
claims subCollection
string claimTopic: (numeric type of claim being added, linked to issuer)
string ID: (Claim's unique ID number)
string data: (Data associated with claim in bytes)
string issuer:(Claim issuer's smart contract address)
string signature:(Message signed by claim issuer, used for validation)
string claimTopic: (numeric type of claim being added, linked to issuer)
string URI:(Reference to off chain claim storage location)
newClaims subCollection
string claimType: (type of claim being added, linked to issuer)
string data: (Data associated with claim in bytes)
string issuer:(Claim issuer's smart contract address)
string signature:(Message signed by claim issuer, used for validation)
string URI:(Reference to off chain claim storage location)
1:1
Issuer Collection
string claimType: (type of claim issued e.g. Driving License)
string contractAddress: (Blockchain Address of Smart Contract)
Session Collection
string sessionID: (sessionID generated by USSD API)
int page:(the last page sent to the clent)
string phone: (hash of client's phone number)
string token: (session authentication token if issued)
Verifier Collection
string phoneNumber: (hash of verifier's phone number)
object account: (Ethereum Account address and encrypted private key)
string idContractAddress: (Blockchain address of ID Smart Contract)
string Name: (Verifierer's Name)
object Password:(Encrypted password)

Figure 4.1: Overview of database. Coloured blocks represent collections. List items represent documents. List headers represent document index.

The Firebase suite contains a built-in authentication module; however, this is designed to run on the client device using the standard Firebase SDK. Since all logic is handled on the server, this application is required to use the Firebase admin SDK, which only provides a method to issue an authentication token.

One of these tokens is issued when the user or verifier logs into the application successfully using their pin code. Once a token is issued, it is stored in the session collection of the database as proof that the user logged in successfully during that session.

4.0.3 Smart Contracts

There are two solidity smart contracts that are used to implement this identity system: Identity.sol and ClaimIssuer.sol. These smart contracts inherit the ERC734 and ERC735 standards, which are described in previous sections.

The identity.sol smart contract contains four key/value mappings (similar to a dictionary in other languages) and methods to add, remove, read, or update the mappings. The key and value types for these mappings are described in Table 4.1 below.

Mapping	Type:index	index	Type:value	value
keys	keccak256 hash of ethereum address	key	struct	key struct
keysByPurpose	uint256	purpose	array_key	all key indexes with given purpose
claims	bytes 32	claimID	struct	claim struct
claimsByTopic	uint256	topic	array_claims	all claim IDs of given topic

Table 4.1: Table that lists the four different key/value mappings stored in the identity smart contract in terms of index(key) and value

The key data structure contains the attributes listed in Table 4.2 below. Any account with a management key for the smart contract can add, remove, or update the keys saved in the smart contract. Any account can read the smart contract.

Attribute	Type	Value
key	bytes32	keccak256 hash of ethereum address
Purpose	uint256	purpose (e.g. 1 for management, 2 for action)
Type	uint256	schema used (e.g. ECDSA)

Table 4.2: Table describing the attributes that make up a key struct in the identity smart contract

The claim data structure mentioned in Table 4.1 above contains the attributes listed in Table 4.3 below. Similarly to keys, any account with a management key for the smart contract is permitted to add new claims, remove, or update existing claims. Any account can read the claims from the smart contract.

Attribute	Type	Value
claimID	bytes32	Unique claim ID
Topic	uint256	The type of claim (e.g. 1 for name)
Issuer	address	Claim issuer's smart contract address
Scheme	uint256	The scheme used to verify this Claim
Signature	bytes32	Signed message of form keccak256(abi.encode(address identityHolder_address, uint256 _ topic, bytes data))
Data	bytes	Data about the claim, encoded as bytes
URI	string	The location of the claim e.g. http link, ipfs hash etc...

Table 4.3: Table describing the attributes that make up a claim data structure in the identity smart contract

The claim issuer smart contract inherits from the identity smart contract, which means that it contains all the same information and methods as an identity smart contract. However, it also contains an additional mapping called revokedClaims. The indexes of this mapping are claim signatures. The signatures correspond to a true or false value.

In addition to this additional mapping, the contract contains methods to check if a claim is valid. The method of verifying whether a claim is valid requires four parameters, as outlined in Table 4.4 below.

Parameter	Type	Value
Identity	bytes32	The identity smart contract the claim is associated with
Topic	uint256	The claims topic
Signature	address	The signature of the claim
Data	uint256	The data field of the claim

Table 4.4: Table describing the parameters required to validate a claim in the smart contract of the claim issuer.

First, the value of revokedClaims corresponding to the claim signature is read. If the claim has been revoked, this value will be false. Then the public key used to sign the claim is recovered. The claim is considered valid if the revokedClaims value is false and if the key used to sign the claim is still stored in the keys mapping and has purpose 3.

Any account can perform this function, meaning that any verifier can verify any claim as long as they know the claim issuer's smart contract address.

4.0.4 JavaScript Programme

The JavaScript programme is the centre of this application and links all other aspects. The app consists of two cloud functions, one for verifiers and one for claimers. These two functions act like page managers; they determine the last page presented to the client and combine that with the text input to perform a function.

The functions performed by these page managers are stored in different files spread across four different sub-folders, as shown in the application diagram in figure 4.4.

Figure 4.2 shows how the page manager functions for a claimer, and 4.3 shows how the page manager functions for a verifier.

Most of the actions carried out by the application rely on Web3.js to interact with the blockchain by calling already deployed smart contracts. In order to access a contract's methods using JavaScript, a JSON application binary interface (ABI) is required. This is stored in the contracts folder for each of the two smart contracts. The ABI lists all events and methods of a smart contract along with their parameters.

When a smart contract's method needs to be called, Web3 creates a contract object in memory using the ABI and the deployed contract's address. A transaction object is then created by calling the desired method. Finally, this transaction is sent to be mined.

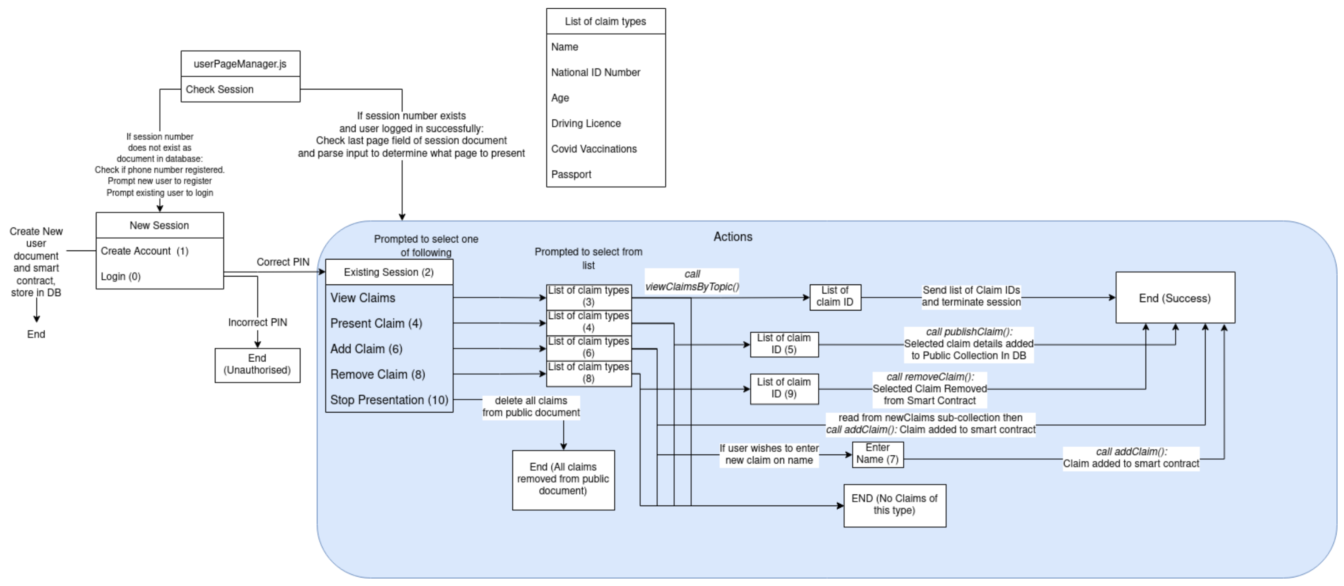

List of claim types
Name
National ID Number
Age
Driving Licence
Covid Vaccinations
Passport
userPageManager.js
Check Session
If session number does not exist as document in database: Check if phone number registered. Prompt new user to register Prompt existing user to login
If session number exists and user logged in successfully: Check last page field of session document and parse input to determine what page to present
Create New user document and smart contract, store in DB
End
New Session
Create Account (1)
Login (0)
Correct PIN
Incorrect PIN
End (Unauthorised)
Actions
Prompted to select one of following
Existing Session (2)
View Claims
Present Claim (4)
Add Claim (6)
Remove Claim (8)
Stop Presentation (10)
Prompted to select from list
List of claim types (3)
List of claim types (4)
List of claim types (6)
List of claim types (8)
call viewClaimsByTopic()
List of claim ID
Send list of Claim IDs and terminate session
End (Success)
List of claim ID (5)
call publishClaim(): Selected claim details added to Public Collection In DB
List of claim ID (9)
call removeClaim(): Selected Claim Removed from Smart Contract
read from newClaims sub-collection then call addClaim(): Claim added to smart contract
If user wishes to enter new claim on name
Enter Name (7)
call addClaim(): Claim added to smart contract
delete all claims from public document
End (All claims removed from public document)
END (No Claims of this type)

Figure 4.2: Application flow when a claimer dials the USSD code. Read from left to right. Numbers in brackets indicate the page number and that the user is expected to input a response. Captions indicate functions called between steps and their result. At end points session data is deleted and the session terminated

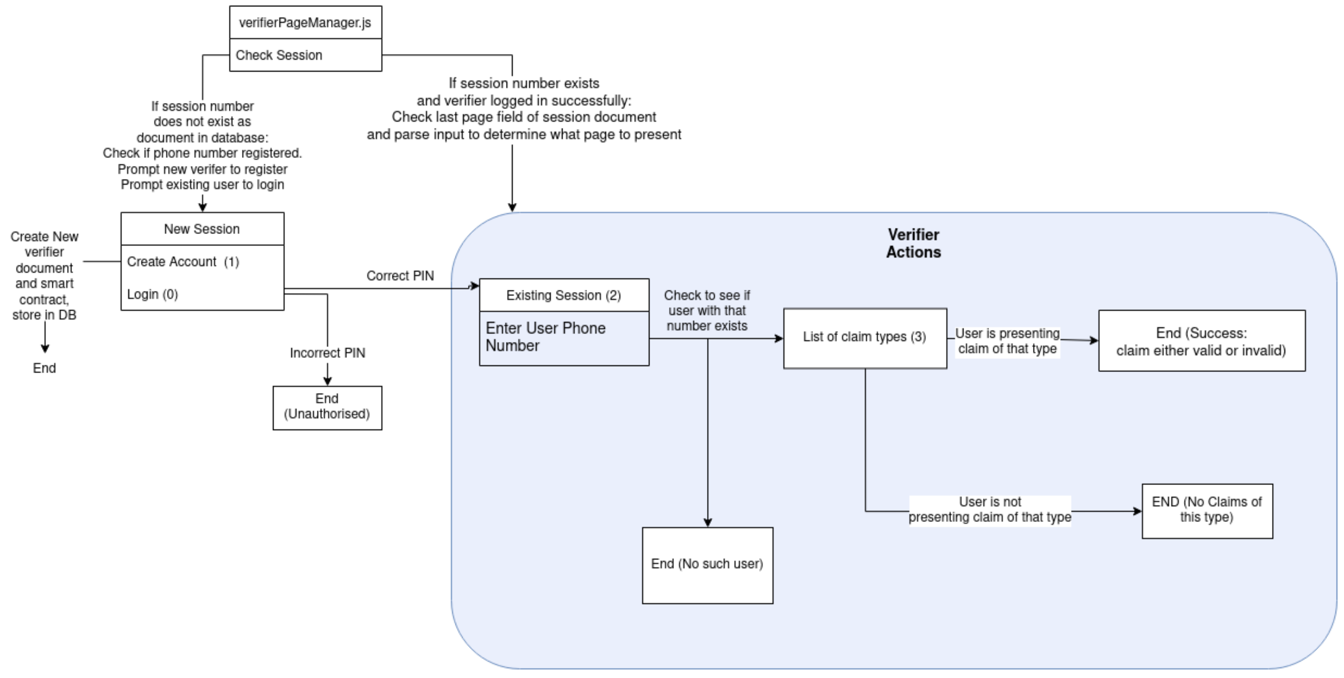

verifierPageManager.js
Check Session
If session number does not exist as document in database: Check if phone number registered. Prompt new verifer to register Prompt existing user to login
If session number exists and verifier logged in successfully: Check last page field of session document and parse input to determine what page to present
Create New verifier document and smart contract, store in DB
End
New Session
Create Account (1)
Login (0)
Correct PIN
Incorrect PIN
End (Unauthorised)
Verifier Actions
Existing Session (2)
Enter User Phone Number
Check to see if user with that number exists
List of claim types (3)
User is presenting claim of that type
End (Success: claim either valid or invalid)
User is not presenting claim of that type
END (No Claims of this type)
End (No such user)

Figure 4.3: Application flow when a verifier dials the USSD code. Read from left to right. Numbers in brackets indicate the page number and that the user is expected to input a response. Captions indicate functions called between steps and their result. At end points session data is deleted and the session terminated

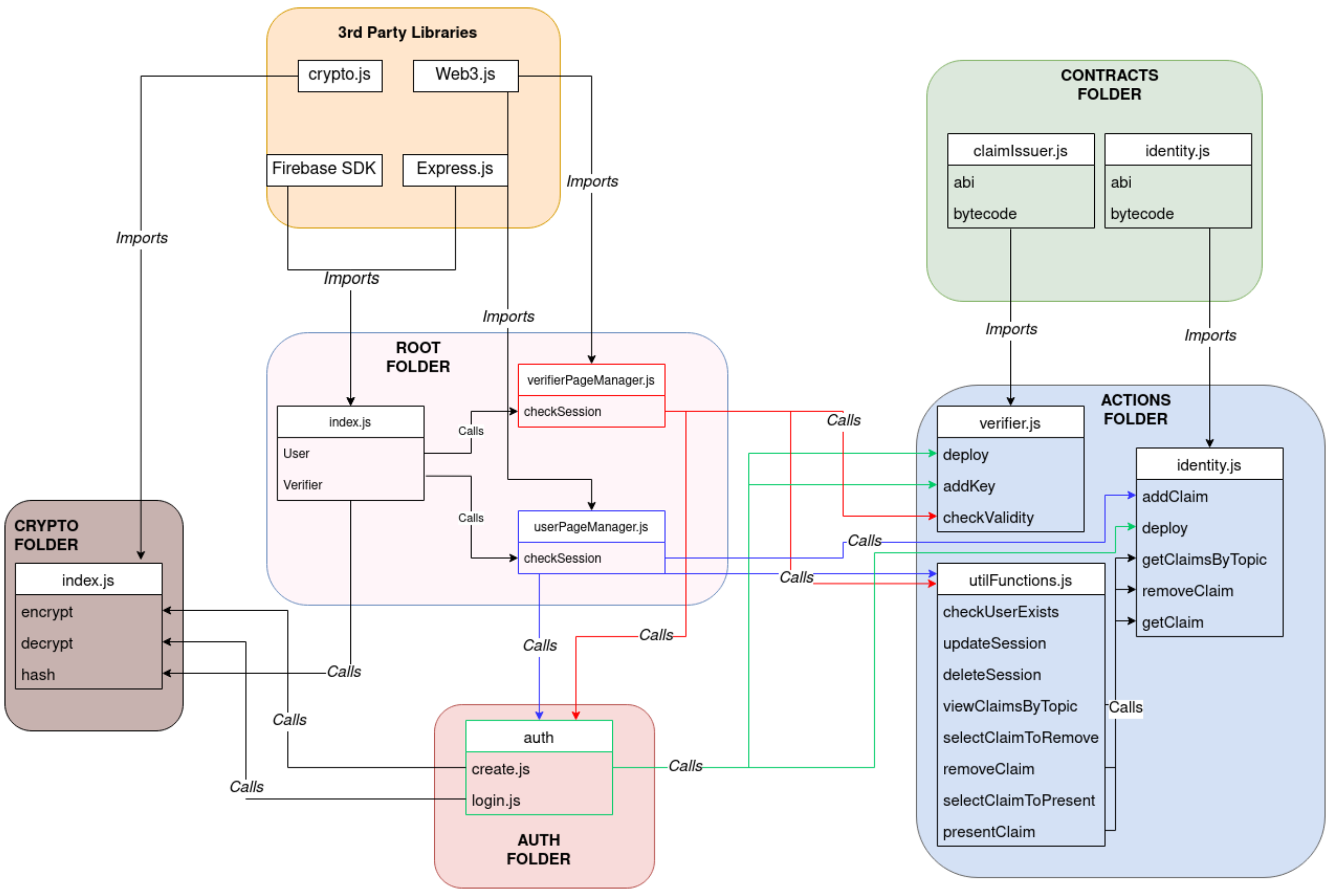


Figure 4.4: Overview of application architecture. Coloured boxes represent folders, lists represent files, list items represent functions. Lines illustrate how the different files interact.

4.1 Summary

This chapter describes how the tools introduced in the previous chapter are implemented to create a working proof of concept that allows users to manage their digital identity through a USSD interface and verifiers to verify claims made by users also exclusively using USSD.

Chapter 5

Evaluation

In this chapter the proof-of-concept application will be demonstrated by presenting scenarios of how to use the application in a real-world situation. The application will then be reviewed, examining it in terms of security, privacy, cost, decentralisation, and utility.

5.1 Use Case: Adding claim

Figure 5.1 shows how a user can add a claim to their identity from the database in as few as five steps. Here, the user was successful, which means that a claim of that type had already been signed and added to its database page. If no claims of this type were present, step five would result in a failed attempt.

5.2 Use Case: Presentation of a claim for inspection

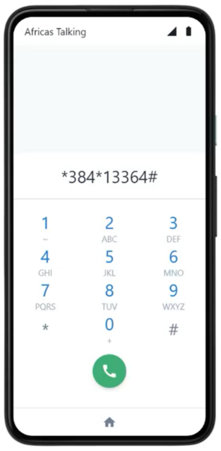
Figure 5.2 shows how a user can add a claim to their identity from the database in as little as five steps.

5.3 Use Case: Verification of a claim

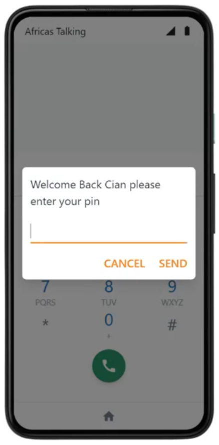
Figure 5.3 shows how a verifier can approve a claim in as little as five steps.

In this case, the user had already selected their driving licence for inspection in 5.2, and the claim was successfully validated. If the claim is invalid, the verifier will receive a "Document Invalid" message at the end of this process. And if the user has not shared a claim of that type, they will get a message saying "No such claim shared".

Figure 5.1: How to add Claim

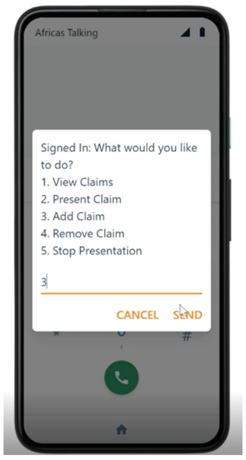


(a) Step One: The user dials the USSD shortcode


Africas Talking

Welcome Back Cian please enter your pin

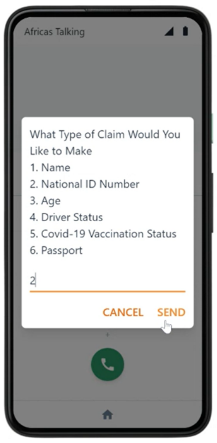
(b) Step Two: The user logs in with their PIN code


Africas Talking

Signed In: What would you like to do?
1. View Claims
2. Present Claim
3. Add Claim
4. Remove Claim
5. Stop Presentation

3

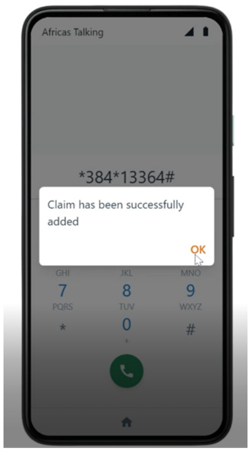
(c) Step Three: The user selects add claim from menu


Africas Talking

What Type of Claim Would You Like to Make
1. Name
2. National ID Number
3. Age
4. Driver Status
5. Covid-19 Vaccination Status
6. Passport

2

(d) Step Four: The user selects the type of claim to add

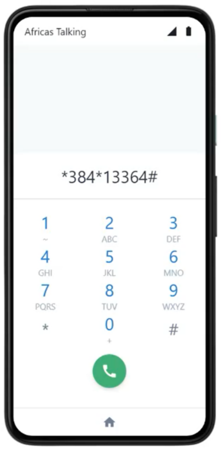

Africas Talking

*384*13364#

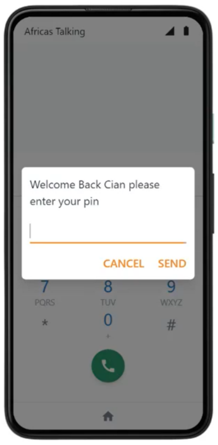
Claim has been successfully added

(e) Step Five: Claim Successfully Added

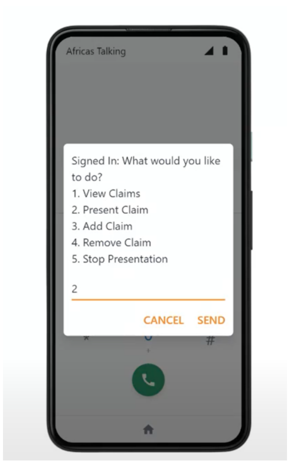
Figure 5.2: How to present claim



(a) Step One: The user dials the USSD shortcode

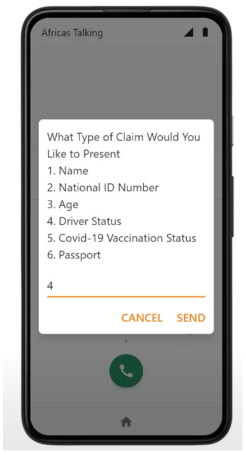

Welcome Back Cian please enter your pin

(b) Step Two: The user logs in with their PIN code


Signed In: What would you like to do?
1. View Claims
2. Present Claim
3. Add Claim
4. Remove Claim
5. Stop Presentation

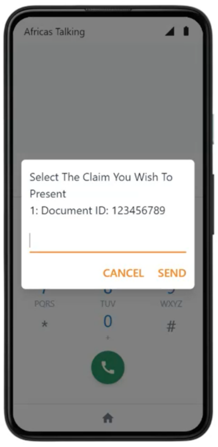
2

(c) Step Three: The user selects present claim from menu

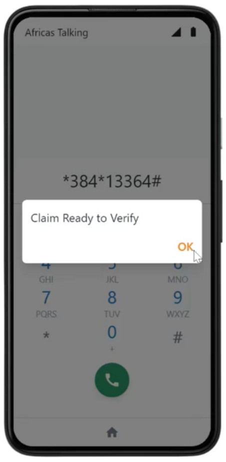

What Type of Claim Would You Like to Present
1. Name
2. National ID Number
3. Age
4. Driver Status
5. Covid-19 Vaccination Status
6. Passport

4

(d) Step Four: The user selects the type of claim to present


Select The Claim You Wish To Present
1: Document ID: 123456789

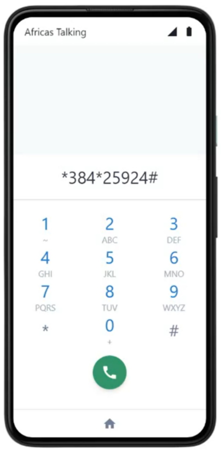
(e) Step Five: The user selects the specific claim to present


*384*13364#

Claim Ready to Verify

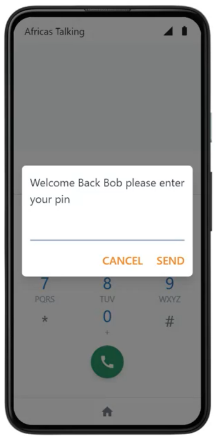
(f) Step Six: Claim Ready for Inspection

Figure 5.3: How to verify a claim



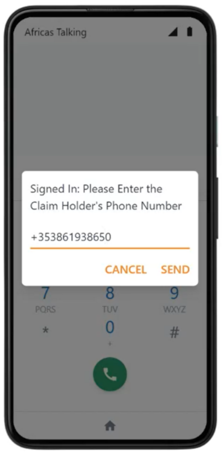
*384*25924#

(a) Step One: The verifier dials the USSD shortcode



Welcome Back Bob please enter your pin

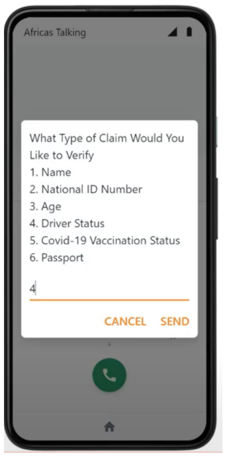
(b) Step Two: The verifier logs in with their PIN code



Signed In: Please Enter the Claim Holder's Phone Number

+353861938650

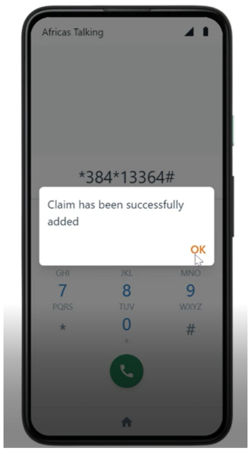
(c) Step Three: The verifier enters the phone number of the user they are inspecting



What Type of Claim Would You Like to Verify
1. Name
2. National ID Number
3. Age
4. Driver Status
5. Covid-19 Vaccination Status
6. Passport

4

(d) Step Four: The verifier selects the type of claim to verify



*384*13364#

Claim has been successfully added

(e) Step Five: Claim is valid

5.4 Security

Security is an important consideration, especially when handling personal data from people. In its current state, the application presented in this paper is not very secure. Users cannot view or change their personal information without entering a valid PIN, but this is the only authentication step present. Furthermore, these PINs are only 6 characters long and contain only numeric characters.

User authentication in general is not very secure, as the authentication tokens issued are currently stored in the cloud, meaning that once a user has entered their pin correctly, the only thing that ensures that it is the same user is the session ID. It is done this way, as the USSD API is not able to add an authorisation header to http requests, as the user is not authenticated when first connecting to the API.

This means that an attacker could theoretically send a http POST request to the endpoint containing a user's phone number and the session ID without knowing their PIN code and be recognised as the user. This is mitigated by deleting the session document whenever a session ends or an error occurs, meaning that an attacker would have to perform any malicious actions while the session is active.

Another issue with PIN codes is that it is not possible to mask the digits that are being entered on a feature phone, meaning that it is possible to steal someone's pin by simply looking over their shoulder while they enter it into the phone.

Data stored in the cloud are encrypted using AES-256 bit encryption as standard by Google and are also encrypted or hashed before storing by the application, meaning that they are essentially doubly encrypted in the cloud storage. However, the information stored is very sensitive, for example, account private keys are stored on the database, so this is necessary. The built-in encryption is also disabled for users with admin privileges, which means that they could be targeted by potential hackers.

Another issue that has been mentioned previously is that data transmitted on GSM networks over USSD are either not encrypted or do not feature strong encryption. This leaves users open to man-in-the-middle attacks from attackers who are listening on the network. Additionally, to connect to this application over USSD users must be connected through a gateway service provided by a telecom network. This is a potential weak point in the flow of information and is fully under the control of the telecom provider.

5.4.1 Possible Improvements

Preventing brute force attacks against users pin codes can be done by setting a limit on the number of incorrect guesses permitted. Preventing users from using weak pins could also improve security and make it harder for an attacker to guess someone's PIN.

Studies have shown that by blacklisting specific pin codes, it is possible to improve the security of mobile phone pins against attacks(**?**). By adding a counter that limits the number of subsequent incorrect guesses possible and incorporating a blacklist of weak PIN codes, it should make pins more difficult to break.

One potential option to make the application more secure is to use a Sim Toolkit (STK) application. Most sim cards are programmable Javacards and can be programmed to run small Java applications known as applets. These applets are accessed by dialling a code similar to USSD. The user can then enter their PIN and have it authenticated by the applet; then the applet calls the USSD code, meaning that only authenticated users can access the USSD api.

This is how the M-Pesa mobile money system in Kenya works, and although it should increase the security of the system, it introduces some additional complexity. First, the applet would need to be installed on the sim card, which means that a physical onboarding process would be required for users. Second, if the phone is lost or stolen, this means that access to the account is also lost. This is probably one of the reasons why telecommunications companies tend to control the mobile money industry in Africa, as they can ensure that their mobile money application is installed on every SIM card they provide.

The storage of sensitive information, such as private keys on the SIM card, was also considered, and there are examples of cryptocurrency wallets that are designed to live on Javacard devices. However, doing this would further increase the risk of losing a phone.

Therefore, with this in mind and given the limited capabilities of the client device, it is necessary to store this information somewhere online. However, one potential improvement would be to create user wallets using the ethers.js library instead of Web3.js. The main benefit of ethers.js over Web3.js is that it automatically generates a mnemonic phrase that can be used to recover the wallet in case the pin is lost.

The issue of weak encryption for data transmitted over GSM networks has been identified by a number of parties. Some researchers have even suggested methods to employ end-to-end encryption from the client device using only sim capabilities(42). Yet again, though, this raises the issue that in order to implement this on the client's device, a programme must be installed on their SIM card. Unfortunately, telecom providers are the best equipped to solve this issue, as they have the capability to implement this security on their SIM cards or on their networks.

5.5 Utility

Currently the application works and it is possible for a user to manage their identity by adding, removing, and updating claims through the USSD interface. Claims can also be verified using USSD. This proves that it is possible to make a functional, general-purpose identity management system using Ethereum and USSD. However, there are areas that have been overlooked and would hinder the application in the real world.

The first of these is that there is currently no interface for claim issuers to add user claims to the database. This was done manually for testing purposes. Without this feature, users would not be able to add new claims to their identity.

The application does what it is supposed to do, which is to provide a way to access a digital identity for those without access to a smartphone. However, what has not been considered in this implementation is a means of exporting the user's identity should they upgrade to a smartphone or even if they have a new phone number.

An obvious flaw in this system is that there is no way to tell if the person using the phone is who they claim to be. With physical credentials, this is accomplished using a photograph, but the text-based interface of USSD does not allow images to be sent. This means that if someone learns another person's PIN code, they would potentially be able to mimic them.

5.5.1 Possible Improvements

Claim issuers should obviously be able to sign claims and issue them to users. A USSD interface would not be the ideal implementation for this task. Information such as the claim data, claimer's contract address, and so on would need to be entered, and doing this on a phone's keypad may lead to errors. If this system were ever used, a web-based interface would be developed for claim issuers to sign claims, allowing them to generate claims simply by filling in some simple menus.

Exporting data is something that should be done with care. If, for example, a user could request to export their account with only their PIN, this could allow attackers to steal people's digital identities permanently. Therefore, for this reason, additional security checks should be performed, perhaps using biometric data as a proof of identity. The actual act of exporting the information would not be difficult. In the event of a phone number change, the new number can be linked to the existing smart contract and the Ethereum account simply by copying the details between documents in the database.

In the case of a user upgrading to a smartphone, the migration is less simple. One option would be to have them download an Ethereum wallet, such as Metamask, on the smartphone and create a new Etherum account using the wallet. The smart contract management key can be changed to the saved smartphone account. Another option would be to develop a smartphone application specifically for use with these smart contracts.

As for the issue of false identification, the main difference between using this system, or using physical identifiers, is that there is no picture provided using USSD. One potential way to overcome this issue would be to use multimedia messages (MMS) to supplement the USSD claim validation process. An image of the claim holder could be stored at the address referenced by the URI field of a claim. Then, when a verifier checks an ID, that image could be sent over MMS using a provider such as Twilio.

5.6 Cost

Cost is an important aspect of this project, as if the system is expensive to use, people will be less likely to use it. Currently, the application is hosted on a test network, which means that no real money is spent; however, if it were deployed on a full blockchain, this would not be the case. Any transactions that alter the blockchain's state (e.g. adding a claim) have a transaction cost. The amount of data changed will increase the cost of the transaction.

With the current setup, deploying an identity smart contract costs roughly 0.12 LXE, and adding a claim costs 0.09 LXE. At the current exchange rate (19 April 2022), this equals $1.99 US to deploy a contract or $1.38 US to add a claim. This is not huge, but it is an additional cost, which could be a problem, particularly in low-income settings.

5.6.1 Possible Improvements

To reduce transaction costs using an Ethereum-based blockchain, the amount of data contained in each transaction should be minimised. For deployment transactions, this means that the smart contracts being deployed should be made smaller by removing unused methods and events. Similarly, when adding a claim to the blockchain, the data field should be kept as small as possible. This could be done by storing the actual claim data off-chain and linking via the URI field. The data field could then be minimised.

Another way to improve these costs for users would be to transfer the system to a permissioned blockchain, such as Hyperledger Indy, which does not have transaction fees. This would also have the effect of increasing the transaction speed, since permissioned blockchains use a simplier consensus method. Alternatively, a more specialised public blockchain, such as Veres One, where the cost of creating a DID is estimated to be $1 US and the cost of updating a DID only $0.25. These costs should also remain fixed with Veres, as nodes in the blockchain are paid directly by the governing body. This means that fluctuations in the price of crypto currencies are not an issue.

5.7 Privacy

Data privacy is an important consideration for any project dealing with personal data of people, and this project is no different. The two main places where user data are stored are in the database, where your blockchain account details are stored along with your phone number, name, and PIN. All information in the database is encrypted or hashed before storage and is encrypted by google. This means that no one can read any of these data without knowing the encryption key or the user's pin code and phone number.

The other place where data is stored is on the blockchain smart contracts, and this is more of an issue, as anyone can read data from a public blockchain. For testing purposes, the claim data was stored as bytes directly in the smart contract, which means that these data could be read by anyone. Another issue with this is that the data stored on the blockchain remains there forever, even if a user removes a claim from their account at a later date, and anyone could go back to an earlier transaction and view those data. A different strategy would need to be employed to improve user privacy.

One further area which could be improved is in data sharing; currently, when a user chooses to present a claim, any verifier can check it while it is being shared, additionally all of the information about that claim is being shared, and the user may wish to only share some of the information.

5.7.1 Possible Solutions

One method to improve the privacy of the data stored on the blockchain would simply be to encrypt it before writing any data. This would certainly make it more protected, but the fact remains that the data remain on the blockchain forever. This means that in a few years the encryption used might be easier to break, and someone can read the data stored there.

Another possibility would be to store the information somewhere off the blockchain, and the smart contract used allows for this, as there is a URI field for each claim, which could point to the off-chain location where the data are held. Data stored off-chain should also be encrypted, but there are many options for where to store it. One of the most popular ways to store and share data using blockchain is IPFS (Interplanetary File System). This is a decentralised file-sharing system where content is stored on one node and broadcasts to other nodes what it has. Files are indexed using a content ID, which is determined by the content of the file. Find a file stored at another node. The nodes host a distributed hash table, which is used to find content stored on other nodes. However, this information is still publicly available, but a private IPFS network could be used, meaning that only authorised nodes can access the stored information.

One of the benefits of using IPFS is that files can be stored in directories and indexed in a Unix-like fashion. This, combined with the encryption keys listed in the smart contract, could be used to limit the amount of information a verifier gets to see. A user could specify who they want to see a specific claim and add their public address as an encryption key, that claim could then be downloaded from IPFS and unencrypted with the user's private key. The specific information they wish to share could then be encrypted using the verifier's public key and stored in the same directory on IPFS with the verifier's public key as the index within the directory.

5.8 Centralization

Much of the literature review and research done for this article was based on the idea of decentralised digital identity and the goal was to create a decentralised system that could be accessed by anyone.

Unfortunately, however, designing for limited client devices, such as feature phones meant that functionality such as data storage and account management had to take place on a centralised cloud platform. Without access to individual phones, it is not feasible to do this in any other way.

Additionally, the designed system relies on a telecommunications network provider to route requests through a gateway that they own, adding another centralised point of failure to the system. Again, this is an unfortunate necessity, as the edge devices for which this system was designed have no means of directly accessing the Internet, and therefore must go through an intermediary.

The ten principles of SSI outlined in Chapter 1 can be used to evaluate the system for decentralisation. The designed system ticks some of the boxes but falls short in some areas. The main areas where the system falls short are...

Transparency; the everyday user cannot see how the system is working behind the scenes, as everything is hosted on the cloud and behind a USSD gateway. Portability; this has been mentioned previously, but users are currently unable to export their data when they choose to. Minimisation; Users cannot specify which aspect of a claim they would like to share, thus minimising data transfer; instead, they must share the whole claim.

5.9 Governance

One aspect of this system that has not yet been discussed is the governance framework that might be used. For this system to work, there must be a set of approved and trusted claim issuers that issue specific predefined claims. Determining who approves of this list of claim issuers would depend largely on the situation in which the system is deployed.

An obvious choice for a governing authority would be the national government of the countries in which the application is deployed. Most nations have official government-run departments that are already in charge of issuing credentials to citizens, so it would make sense to allow these offices to issue and sign digital claims.However, governments are not always stable or trustworthy.

In these situations, people are forced to flee their homes as refugees. On 6 April 2021, there were more than 6.6 million refugees living in camps around the world(47). These people could receive digital identities to help gain access to food or medical supplies. In this kind of setting, a charity or nongovernmental organisation would probably be the best choice to govern an identity system like this and determine who can issue claims.

Regardless of who is in control, it is crucial that they have the best interests of all users in mind and work to ensure that the system is used for the benefit of those who need it.

Chapter 6

Conclusions & Future Work

Self sovereign identity is still a new and developing field of work, and as such there is no agreed upon best practise for defining a decentralised identity system. Building one of these systems to work on feature phones is even less so. This is highlighted by the many different approaches discussed in Chapter 2. The variety of different standards and methods which have been proposed ensure that the capabilities of any two systems can differ greatly, and it is up to the designer to decide what features should be included or excluded.

Integrating a new and emerging technology, such as SSI and an older technology, such as USSD, is not an area with much ongoing research; however, it is a topic which appears to have potential to improve people's lives. USSD powered mobile money applications have proven to lower barriers to financial inclusion in Sub-Saharan Africa and are used by thousands of people every day to pay peers or receive financial support.

Three different projects that used blockchain ledgers to offer services to people without access to smartphones were examined when designing this application. All three take a slightly different approach to user account management. Kiva links user IDs to their national identity card and holds keys under guardianship for users without access to the smartphone, allowing them to check their keys when needed. Leaf linked users' mobile money accounts to their blockchain account, allowing them to send funds peer-to-peer across borders with much lower transaction fees than with competitors. And WFP Building Blocks held users' keys fully under guardianship and signed transactions on their behalf.

These case studies showed two things: that storing credentials and account details directly on the users' phones is not viable, and that people are willing to trust these organisations to store their keys for them.

With this knowledge, it was clear that designing a digital identity system for use on a feature phone would require three different elements: A USSD API provider, a decentralised identity management system, and a database to store user's keys until they required them.

The Etherum blockchain infrastructure was decided as the verifiable data register to store DIDs, and the ERC734 and ERC735 smart contract standards were chosen as the basis for the identity system. These two standards combine to create a general purpose key and claim manager which allow verified claims about an identity to be added to a smart contract representing that identity.

These standards are slightly outdated now, having been proposed in 2017, but have proven to be effective in creating a general purpose identity management system. Using these standards, it was possible to create a smart contract of the claim issuer for issuing and verifying claims about other identities and an identity smart contract to store claim information about that identity.

One of the benefits of using these contracts is that they are extremely flexible, any account can create an identity contract and start managing claims about themselves. Similarly, any account can create a smart contract with the claim issuer and start issuing claims. This means that the identity system could be adapted to suit a wide range of needs.

However, it also means that users must be careful and ensure that they are only interacting with certified claim issuers. For this reason, and because keys are being stored on behalf of users, it is important that governance of the system is managed by a trusted entity such as a government or non-governmental organisation depending on the setting.

It is my opinion that this system would be best suited to issue emergency digital identities to people in need, such as refugees, so that they can gain access to essentials such as food or medicines, similar to the Building Blocks example. Users could add additional claims to their smart identity contract to prove, for example, that they require a specific drug. Then, when such resources are being distributed, the distributors could verify users' claims to confirm that they require special aid.

The system could also be used for day-to-day identification in non-emergency situations, such as proving that you have a drivers licence, however physical credentials are likely still better suited to this task. Primarily because there is no way to determine if the person using a phone is actually the person registered under that number. For example, with the current system one could theoretically lend their phone to a younger family member and share the pin code. The phone could then be used to ”prove” that they are over 18 years of age and allow them to purchase alcohol illegally. Therefore, for this reason, this system appears to be best suited for issuing emergency digital identities.

6.1 Future Work

The proof of concept application built as part of this project has a number of shortcomings, which are outlined in Chapter 5. Any future work on this project should prioritise addressing these issues.

The most important issues to address are those related to privacy. To make the storage of claims more private and only allow authorised users to read them, a lot of work needs to be done. In Chapter 5 IPFS is suggested as an off-chain decentralised data store which could be used to host user claims. However, the viability of this would need to be determined, and perhaps a better solution exists.

Another possibility would be to allow users to transfer funds to peers using their blockchain accounts. Since the system already provides users with the means to sign and send transactions, it should not be difficult to implement this feature. However, if users start to store real cryptocurrency with market value in their accounts, then the database that stores their account details may become attractive to attackers looking to steal that money.

Ideally given enough development time, this system would be fully portable so that users can export their identity to new devices easily; privacy-preserving, only allowing specified accounts to read data from a user smart contract, and allowing users to revoke that access at any time. However, even if this is achieved, I still believe that this system should only be used in emergency scenarios or as a temporary service for users who do not yet have smartphones.

Designing the system for such basic edge devices means that compromises must be made somewhere, and unfortunately these compromises mean that the system is not fully decentralised and is not as secure as an SSI system designed for use with a smartphone.

Bibliography

[1] Abbeel, P., and Lee, E. Dtmf signaling, 2011. Available at https://ptolemy.berkeley.edu/eecs20/week2/dtmf.html, Accessed 13 April 2022.

[2] Africa's Talking. Africa's Talking API reference, 2022. Available at https://developers.africastalking.com/, Accessed 13 April 2022.

[3] Allen, C. The Path to Self-Sovereign Identity, Apr. 2016. Available at https://www.coindesk.com/markets/2016/04/27/the-path-to-self-sovereign-identity/, accessed 12 April 2022.

[4] Becze, M., Jameson, H., and et al. Eip-20: Token standard, November 2015. Available at https://eips.ethereum.org/EIPS/eip-20, Accessed 5 April 2022.

[5] Boudj, A. Onchainid github repository, 2021. Available at https://github.com/onchain-id, Accessed 13 April 2022.

[6] Buterin, V. Ethereum whitepaper, 2014. Available at https://ethereum.org/en/whitepaper/.

[7] Buterin, V. A prehistory of the Ethereum Protocol, 2017. Available at https://vitalik.ca/general/2017/09/14/prehistory.html.

[8] Cedex, S. A. ETSI TS 122 030 v10.0.0 (2011-05)", 2011. Available at https://www.etsi.org/deliver/etsi_ts/122000_122099/122030/10.00.00_60/ts_122030v100000p.pdf, Accessed 4 April 2022.

[9] Collins English Dictionary. Collins english dictionary, identity, 2022. Available at https://www.collinsdictionary.com/dictionary/english/identity, Accessed: Mar 30 2021.

[10] Department of Social Protection. Get a personal public service (pps) number, 2021. Available at https://www.gov.ie/en/service/12e6de-get-a-personal-public-service-pps-number/, Accessed: Mar 30 2021.

[11] Doffman, Z. Chinese hackers just gave us all a reason to stop sending SMS messages, 2019. "Available at https://www.forbes.com/sites/zakdoffman/2019/11/03/chinese-hackers-just-gave-us-all-a-reason-to-stop-sending-sms-messages/, accessed "April 11 2022".

[12] e-Estonia. X-Road, 2018. "Available at https://e-estonia.com/solutions/interoperability-services/x-road/, accessed "April 11 2022".

[13] Ethereum Foundation. Ethereum improvement proposals, 2022. Available at https://eips.ethereum.org/, Accessed 5 April 2022.

[14] Ethereum Foundation. Web3.js Documentation, 2022. Available at https://web3js.readthedocs.io/, Accesssed 13 April 2022.

[15] Ethereum Foundation. Web3.js Documentation web3.eth.accounts.recover, 2022. Available at https://web3js.readthedocs.io/, Accesssed 13 April 2022.

[16] European Commission. Eu digital covid certificate, 2021. Available at https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_en, Accessed 14/04/2022.

[17] Evernym. Evernym | The Self-Sovereign Identity Company, 2022. "Available at https://www.evernym.com/, accessed "April 11 2022".

[18] Fintech, L. G. Leaf 2021 Year in Review, Dec. 2021. Available at https://leafglobal.medium.com/leaf-2021-year-in-review-3bac21243d07, Accessed 14/04/2022.

[19] Foundation, H. Hyperledger welcomes project indy. Available at https://www.hyperledger.org/blog/2017/05/02/hyperledger-welcomes-project-indy, accessed 12 April 2022.

[20] Gilbert, P. Strong mobile growth predicted for sub-Saharan Africa – GSMA, September 2020. Available at https://www.connectingafrica.com/author.asp?section_id=761&doc_id=764310, Accessed: Apr 4 2021.

[21] Google. Firebase Documentation, 2022. Available at https://firebase.google.com/docs, Accessed 13 April 2022.

[22] HYPERLEDGER FOUNDATION. Hyperledger indy, 2022. Available at https://www.hyperledger.org/use/hyperledger-indy, accessed 12 April 2022.

[23] KERI PROTOCOL. Key Event Receipt Infrastructure (keri) is the first truly fully decentralized identity system., 2022. "Available at https://keri.one/keri-resources/, accessed "April 11 2022".

[24] KILT PROTOCOL. A blockchain identity protocol for issuing self-sovereign, anonymous, verifiable credentials and decentralized identifiers, 2022. "Available at https://www.kilt.io, accessed "April 11 2022".

[25] KIVA PROTOCOL. Open-source infrastructure for financial inclusion, 2022. Available at https://kivaprotocol.com, accessed 12 April 2022.

[26] LUKSO. Lukso developer documentation, 2022. Available at https://docs.lukso.tech/, Accessed 13 April 2022.

[27] MOTAZ. What is ussd – mmi code – ussd code?, 2018. Available at https://www.cspsprotocol.com/ussd/.

[28] NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system, 2008. Available at https://bitcoin.org/bitcoin.pdf.

[29] NETLER, K., SAUL, H. N., AND BYERS, B. Internet identity workshop 20 book of proceedings, 2015. Available at https://github.com/windley/IIW_homepage/raw/gh-pages/assets/proceedings/IIWXX_Book_of_Proceedings.pdf, Accessed: Mar 30 2021.

[30] ONCHAINID. Decentralized identity system for compliant digital assets, 2022. Available at https://www.onchainid.com/, Accessed 13 April 2022.

[31] OPENJS FOUNDATION. Express js, 2022. Available at https://expressjs.com/, Accessed 13 April 2022.

[32] OPENJS FOUNDATION. Node js, 2022. Available at https://nodejs.org/en/, Accessed 13 April 2022.

[33] POLKADOT. Polkadot: Decentralized web 3.0 blockchain interoperability platform, 2022. "Available at https://polkadot.network/, accessed "April 11 2022".

[34] POSTMAN INC. Postman API platform, 2022. Available at https://www.postman.com/, Accessed 13 April 2022.

[35] Preukschat, A., and Reed, D. *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials.* Manning, June 2021. Google-Books-ID: Nh4uEAAAQBAJ.

[36] Shapshak, T. Mobile money in africa reaches nearly $500bn during pandemic, 2021. Available at https://www.forbes.com/sites/tobyshapshak/2021/05/19/mobile-money-in-africa-reaches-nearly-500bn-during-pandemic/?sh=6f1d25973493,Accessed4April2022.

[37] Silver, L., and Johnson, C. Majorities in sub-Saharan Africa own mobile phones, but smartphone adoption is modest, 2018. Available at https://www.pewresearch.org/global/2018/10/09/majorities-in-sub-saharan-africa-own-mobile-phones-but-smartphone-adoption-is-
Accessed: Apr 4 2021.

[38] Smith, S. Keri papers, Mar. 2022. "Available at https://github.com/SmithSamuelM/Papers/blob/f9cae6553e28687a97833b6874123c16d73f4cd8/whitepapers/KERI_WP_2.x.web.pdf, accessed "April 11 2022".

[39] Sovrin Foundation. Sovrin Foundation homepage, 2022. "Available at https://sovrin.org/, accessed "April 11 2022".

[40] Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., and Allen, C. Decentralized Identifiers (dids) v1.0 core architecture, data model, and representations, 2021. Available at https://www.w3.org/TR/did-core/#introduction, Accessed: Mar 30 2021.

[41] Steele, O., and Sporny, M. Did specification registries. Available at https://www.w3.org/TR/did-spec-registries/#did-methods, Accessed: Mar 30 2021.

[42] Telecommunication Standardization Sector of ITU . Low resource requirement, quantum resistant, encryption of ussd messages for use in financial services. Tech. rep., International Telecommunication Union, December 2021. Available https://www.itu.int/dms_pub/itu-t/opb/tut/T-TUT-PROTO-2021-PDF-E.pdf,, Accessed 12 April 2022.

[43] The World Bank. Identification for Development (ID4D) Global Dataset. Tech. rep., 2018. Available at https://datacatalog.worldbank.org/search/dataset/0040787/Identification-for-Development--ID4D--Global-Dataset, Accessed 19 April 2022.

[44] Three Ireland (Hutchison) Limited. pay as you go price guides.

[45] Truffle Suite. Ganache: One click blockchain by Truffle Suite, 2022. Available at https://trufflesuite.com/ganache/, Accessed 13 April 2022.

[46] Twilio. Twilio communication APIs for SMS, Voice, Video & Authentication, 2022. Available at https://www.twilio.com/, accessed 11 April 2022.

[47] UN Refugee Agency. Refugee camps explained. Available at https://www.unrefugees.org/news/refugee-camps-explained/, Accessed 19 April 2022.

[48] Veres One. Accelerators - Veres One, 2022. "Available at https://veres.one/network/accelerators/, accessed "April 11 2022".

[49] Veres.one. A Globally Interoperable Network for Identity, 2022. "Available at https://veres.one/, accessed "April 11 2022".

[50] Vogelsteller, F. Erc: Key manager 734, October 2017. available at https://github.com/ethereum/eips/issues/734, accessed 5 April 2022.

[51] Vogelsteller, F., and Buterin, V. Eip-1: Eip purpose and guidelines, October 2015. available at https://eips.ethereum.org/EIPS/eip-20, accessed 5 April 2022.

[52] Vogelsteller, F., and Yasaka, T. Eip-725: General key-value store and execution standard, October 2017. available at https://eips.ethereum.org/EIPS/eip-725, accessed 5 April 2022.

[53] von See, A. Mobile money in africa - statistics facts, 2022. Available at https://www.statista.com/topics/6770/mobile-money-in-africa/#dossierKeyfigures, Accessed 4 April 2022.

[54] Wang, F., and De Filippi, P. Self-sovereign identity in a globalized world: Credentials-based identity systems as a driver for economic inclusion. *Frontiers in Blockchain 2* (2020).

[55] World Food Programme. Building Blocks: Blockchain network for humanitarian assistance, 2022. Available at https://innovation.wfp.org/project/building-blocks, Accessed 19 April 2022.

[56] World Food Programme. World Food Programme: Mission, 2022. Available at https://www.wfp.org/overview, Accessed 19 April 2022.

[57] YCharts". Average transaction cost on ethereum network over time, 2022. Available at `https://ycharts.com/indicators/ethereum_average_transaction_fee`, accessed 12 April 2022.

Appendix

1. The codebase containing an implementation of the SSI system described in this paper can be found at `https://github.com/CianMurph/USSiD`
2. A video demonstration of the functioing application can be seen at `https://youtu.be/G-M1xKAR1Ac`