How to build a self-sovereign identity system that is beneficial to both the individual and business

Jothi Moodley

A dissertation submitted to the Faculty of Commerce, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Philosophy.

February 10, 2019

MPhil in Financial Technology,
University of Cape Town

Under the supervision of Associate Professor Co-Pierre Georg, University of Cape Town

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy to the University of Cape Town. It has not before been submitted for any degree or examination.

J.M.

February 10, 2019

Abstract

Self-sovereign identity defines a system in which an entity can generate and maintain their own proof of identity. There are several solutions aimed at providing this service and storing the relevant information on a blockchain. We describe how to develop such a system using Ethereum's smart contract platform and a browser-based application, and we demonstrate its use in a corporate that sells more than one funeral insurance product. Individuals and organizations should be able to create claims on their identities, however, only reputable organizations can verify these claims. These operations are executed by functions contained in the smart contracts and the transactions can be stored on a blockchain. A major benefit of this innovation is that an identity can be easily re-used and we show how an insurance department can do this using credentials already requested by another department. This method allows for much needed efficiency over the current system.

Acknowledgements

I would like to thank the following individuals and organizations.

- My supervisor Associate Professor Co-Pierre Georg for the continuous and valuable feedback.
- Anushka Soma-Patel for guidance and proposing the topic.
- My employer for financial assistance.
- My manager for his patience and allowing me the time to pursue this research.
- The University of Cape Town and the African Institute of Financial Markets and Risk Management for administrative support and resources.
- My mother and colleagues for their patience and support.

Contents

List of figures

1. Introduction

First we consider the characteristics of the current identity model and then explain the new model of self-sovereign identity.

1.1 Current identity system

Identification is a record of information about an individual that can be used to prove a particular characteristic about them such as their age, their right to drive or their ownership of property (Higgs, 2011). At the times of birth and death we are issued with birth and death certificates, respectively. During that lifetime there are numerous forms of identification that are essential for one to access public and financial services and to purchase assets and certain goods (Higgs, 2011). Currently in South Africa, the primary object for proving one's identity is a barcoded green book or smart card with a unique identity number that represents the identity of the holder (Department of Home Affairs, 2018a). This document is issued by the Department of Home Affairs (2018a), a central government authority which citizens and organizations rely on to issue and store correct records. This document serves as the basis from which other identifications can be obtained for other purposes. Some common examples are:

- A driver license allows one the right to drive while a vehicle license allows one the right to drive a specific vehicle.
- Passport and visas enable one to travel outside the country they reside in.
- Property deeds, rental statements and utility bills obtained from landlords and government can serve as proof of residential address (Financial Intelligence Centre, 2005).
- Payslips from an employer are used as proof of income.
- A death report and certificate are required for a beneficiary to receive a pay-out from a funeral cover (Department of Home Affairs, 2018b).

The creation of trust is the main advantage in this model (Birch, 2007). When an individual uses their credential to access a service, the service provider checks that the party that issued the credential is one that they trust. Identification documents are usually in a standardized form (Sovrin Foundation, 2018a) which enables them to be easily recognized and accepted by those who rely on them. They are portable and allow owners access to opportunities (Higgs, 2011). Thus, identification is clearly a useful part of society. However, if original documents are lost or expired, the individual has to re-apply for the credential from the issuer, which is a time-consuming process.

Financial service providers need to follow certain regulations where a customer must be correctly identified through verified documentation. This process is referred to as Know Your Customer (KYC) and in South Africa it is specified under the Financial Intelligence Centre Act (2001). The purpose of this law is to protect customers and businesses from identity fraud

and money laundering (Cox, 2014). The minimum documents required are usually an identity document and a recent proof of residential address. Since a business must be able to prove their compliance with this law for every customer, copies of their data are stored as proof. Also, since an up-to-date verification is required, the process is repeated every time a customer interacts with a provider.

The use of digital services is continually increasing (World Economic Forum, 2018). While this platform offers quicker access to products and services, it has resulted in users inconveniently managing too many accounts with login credentials (Tobin and Reed, 2017; Birch, 2007). Moreover, personal details are saved in these accounts. Digital identity includes electronic records of paper identifications as well as new types of personal information captured by various organizations, such as credit payment profiles (National Credit Regulator, 2007), social networking habits (Russell, 2014) and biometric data (Higgs, 2011). Thus, a user's data is located in multiple databases across all their service providers (Tobin and Reed, 2017). Even within a large corporation, different departments request the same identity information and store it in their own databases (Birch, 2007). Tobin and Reed (2017) explain that this is costly to service providers who need to obtain storage space and provide adequate security to protect their users' data. It is also a disadvantage to individuals as they have less control of the privacy of their data and who it may be shared with. In some cases data can be requested and stored that is not even required at that time (Higgs, 2011). Users may also share data just to access a service, thereby limiting their privacy for convenience.

Even with security measures there have been cases where databases are breached and identity information is stolen (Edwards et al., 2016). With large amounts of consumer data contained in specific locations, a single data breach has a significant impact on users' data privacy and safety (Tobin and Reed, 2017). There are more recent regulations in South Africa that aim to protect the privacy of customers' personal data and how it gets stored and shared (Protection of Personal Information Act, 2013). This adds further cost to the business for securing information and designing processes to comply with this law.

The current identity model is "centralized" (Tobin and Reed, 2017) since the data is issued and managed by governments and corporations. The data risks and inefficiencies of duplicated storage associated with this model motivate the need for a better identity system.

1.2 Self-sovereign identity

Self-sovereign identity (SSI) is a method in which a user can generate and maintain their own identity information such that its meaning can be shared with others without revealing the content of the information (Tobin and Reed, 2017). For example, if one needs to prove to a traffic officer that they have a license to drive, they would only need to show a claim that they have this right and that it is verified by a relevant government department without needing to share further details such as their identity number or date of birth. A good digital identity system should at the least allow any person to create an identity, be useful to users in

verifying themselves and their right to services, must keep user's information private and secure, and ensure that users have control in how their data gets shared with service providers (World Economic Forum, 2018). Self-sovereign identity aims to satisfy these properties (Tobin and Reed, 2017) and is a significant improvement on the current identity model.

This type of system has been made possible with the invention of blockchain technology (Tobin and Reed, 2017; Dunphy and Petitcolas, 2018), a tool that has become widely recognized through cryptocurrencies such as Bitcoin (Rothstein, 2017). A blockchain (Swan, 2015; Rothstein, 2017) is a digital historic record of information that is contained and extended by a set of connected computers called nodes. It uses cryptographic rules to add information in groups of a certain size such that it aligns with the previous information and such that older information cannot be edited or removed. This process is referred to as mining and a node uses electronic resources to compete against other nodes to find a valid group of data. When such a group is found, the nodes check that it is correct and update their records. Since finding the solution requires a certain amount of time, it is difficult for a node to amend previous data. It is meant to be maintained by all nodes rather than one or a few entities, and each node contains the same record of data, also known as a ledger. This concept of a decentralized network that is cryptographically secure enough and difficult to change provides a strong structure for managing identity in a self-sovereign manner (Tobin and Reed, 2017).

A blockchain can also store smart contracts (Swan, 2015; Modi, 2018) that perform tasks automatically when particular conditions are satisfied. Ethereum (Wood, 2014; Buterin, 2014) is a well-known example of this. If an organization makes a claim about a user, a smart contract can enable this operation (Lundkvist et al., 2016), and the information can be encrypted and recorded as a transaction on the ledger where it can be later checked as proof that it occurred. The claim does not need to contain the user's personal data, but just a description of its meaning and that it is verified by the issuer. This functionality explains how blockchain and smart contracts can establish self-sovereign identity. Furthermore, blockchains can be designed (Gatteschi et al., 2018) so that either (i) any node can access and append information, (ii) any node can access information but only specific users can append information, or (iii) they can be completely private where only accepted users can view and add data.

There are several providers that aim to offer self-sovereign identity management applications (see, for example, Mesropyan (2017)), however, they may not all be properly decentralized. Shocard, for example, still retains centralized elements of the current system as a third party provides a server to exchange claims between entities (Grüner et al., 2018). The solutions that appear more viable (Dunphy and Petitcolas, 2018) are uPort (Lundkvist et al., 2016) and Sovrin (Tobin and Reed, 2017; Sovrin Foundation, 2018a) and we examine them in the next chapter. We also review Civic (Lingham and Smith, 2018) as it emphasizes identity verification for KYC purposes. The main features of these systems are that users can create

claims that they have certain attributes, obtain verification of their claims from relevant authorities and share their claims with service providers.

The first usage of self-sovereign identity by a government has been carried out by the City of Zug in Switzerland (Offerman, 2018) with the uPort platform, which uses the public Ethereum blockchain. The City has an identity on the network and can issue citizens who are uPort users with a verified claim that they are a citizen. Citizens can then access other government services using their valid uPort credentials. Sovrin (Tobin and Reed, 2017) offers a semi-private system where only approved reliable organizations represent the nodes, which may be a more appealing solution to companies and regulators as it prevents an untrusted group from controlling more than half the network through the mining process (Tapscott and Tapscott, 2016). Many respected institutions such as Absa, IBM and Cisco have expressed interest in Sovrin (Sovrin Foundation, 2018b).

1.3 Research objective

The objective of this research is to build a self-sovereign identity system with a focus on improving identity verification in an insurance company with two departments, Youth Markets and Family Markets, that offer different funeral policies. Note that this use case is not based on any specific organization, but is generalized from the inefficiencies previously discussed. We consider funeral insurance due to its importance in South Africa (Roth, 2000).

In chapter 2 we review key solutions to understand the various components of self-sovereign identity and explain how funeral insurance works. In chapter 3 we formulate the use case, motivate it and design a system to demonstrate it using Ethereum smart contracts and a decentralized application. A customer first applies for a Single Funeral Plan from Youth Markets using claims on his national identity, residential address and age to prove his eligibility for the product, without supplying any personal data. These claims, verified by government and shared with the insurer, are later re-used when the customer applies for a Family Funeral Plan from Family Markets for himself and a dependent. Claims on his products are also issued to the customer by the insurer. We construct smart contract identities with operations to create, verify and share these claims. Users are represented by cryptographic identifiers in the system and connect to their identities from the application through a wallet. A public ledger records the identity transactions.

We illustrate the use case and comment on the properties of our system in chapter 4. The strengths of our system are that the user has more control of their data and the insurer saves on data storage and security costs. Limitations of our system are that the transactions are not private and there is no tool for the customer to manage his data. We also discuss the general impact SSI systems can be expected to have on both the individual and business.

2. Literature review

2.1 Self-sovereign identity background

The development of self-sovereign identity is motivated by the need for individuals to gain more control and privacy in how their personal information is shared to access services and opportunities (Tobin and Reed, 2017). The idea of users communicating across a digital network in a private and reliable manner has been studied extensively in cryptographic systems with Diffie and Hellman (1979) providing a guide to the early literature. Information is encoded and decoded between two users and initially systems achieved this with one key physically shared by the two parties. However, the management of a single private key is inefficient. Diffie and Hellman (1976) introduced public-key cryptography in which each user has a pair of keys, a public key that can be used by anyone to encode a message and a private key kept only by the receiver to decode the message. Moreover, the message can be digitally signed by the sender with his private key and his public key can be used by the receiver to verify that he is the source. Since it is difficult to determine the private key from the public key, the system is reasonably secure. Pretty Good Privacy (PGP) is a software tool that enables the use of this cryptographic system for general individuals (Garfinkel, 1995). Here a public key is linked to a user's email address and its reliability depends on its verification by other peers in the network. It is considered an example of a decentralized "web of trust" (Richters and Peixoto, 2011). Public-key cryptography is an important part of self-sovereign identity (Lundkvist et al., 2016).

During the progress from centralized identity managed by organizations to self-sovereign identity managed by identity owners, other identity systems have been created: federated and user-centric systems (Tobin and Reed, 2017). With federated identity management, access credentials from one party can be re-used across multiple other service providers (Birch, 2007). While this model reduces the need for users to manage many login details (Maler and Reed, 2008), there are still providers that have a significant amount of control over identities. The user-centric identity model extends the federated model so that individuals can specify which credentials to use to access a service and have more control over how their data is shared (Ahn et al., 2009). However, this system is also dependent on external identity providers (Tobin and Reed, 2017).

The increased interest in blockchain technology has led to recent efforts in developing a system for self-sovereign identity (Dunphy and Petitcolas, 2018). Allen (2016) defines certain principles that a self-sovereign identity system should satisfy. A user must be able to create a unique digital identity for themselves ("existence") with an unlimited duration ("persistence") and be able to remove this identity at any time. A user can create claims on his attributes or be issued with claims from service providers. All claims and supporting data must be owned by the user and available at all times ("access"). An identity is managed by its owner ("control") who has full control over the sharing of his claims with other parties

("consent") and should be able to provide only the least amount of information for a request ("minimalization"). These elements increase privacy for the user. For a user to benefit from a variety of opportunities, their identity must also be "portable" and "widely usable". It must not be limited by a user's location or any provider. Finally, the system should be designed in a "transparent" way with a high priority on user "protection".

2.2 Self-sovereign identity solutions

A number of SSI systems have been recently proposed of which uPort (Lundkvist et al., 2016), Sovrin (Tobin and Reed, 2017; Sovrin Foundation, 2018a) and Civic (Lingham and Smith, 2018) appear to be the most popular. We review those systems here, describing their main characteristics, how they differ and how an organization can integrate it with their current system.

2.2.1 uPort

uPort (Lundkvist et al., 2016) is a self-sovereign identity system that is enabled by the public blockchain Ethereum and its smart contract technology. An identity can be represented by a person, an organisation or even an object and is defined by a public address stored in a smart contract called a "proxy contract". This address is a hexadecimal string of size twenty bytes that uniquely identifies the identity. Hexadecimal refers to a system with sixteen values represented by 0 to 9 and a to f (Encyclopaedia Brittanica, 2016). This address does not need to change if an account is lost and then recovered, thus it is a persistent and immutable identifier. A user uses a mobile application to create and interact with their identity. The public address is linked to a private key that is stored on the mobile device in a secure component. This ensures that only the holder of an account can control the identity contract. However, the account holder is still responsible for managing the private key. Lundkvist et al. (2016) explain that there are other types of contracts created with the identity called the "controller" and "recovery" contracts and these provide most of the functionality. The main function of the proxy contract is to ensure that any operation on the identity can only be executed by the identity owner.

With the mobile application, the user with the correct private key interacts with their controller contract to access their proxy contract. This contract has functions to interact with other applications with permission. It can specify more complex operations that uses the proxy contract to verify the sender's identity and run the operation such as sending a transaction to another uPort identity or other Ethereum address. Attributes describe information about the identity and attestations are attributes that are signed by a relevant authority. A user can create attributes themselves and request a signature of verification from another party or other parties can issue attestations directly to the user. This information can be encrypted for privacy. Other operations that are available through uPort include a service provider requesting a verified claim from a user and a user sharing a claim with a requestor. The controller contract also contains details of the user's recovery contract.

Lundkvist et al. (2016) explain how a recovery contract assists a user in gaining access to their identity contract in the case that their private key is lost or stolen. When an identity is created, a user should specify a group of other users that can be trusted to recover the victim's identity if necessary. More than half of these recovery delegates need to provide their signatures for the recovery to be successful. The recovery contract replaces the controller address with a new address and the controller contract is updated without affecting the user's proxy contract. This update takes a minimum amount of time before it is properly complete. This provides security since if one tries to steal the identity using a stolen key, the owner can still recover their identity in time. The recovery contract includes operations for a delegate to sign to update the user's address and for a user to update their delegates. Updating of delegates also requires a minimum amount of time, which helps prevent one from updating delegates with untrusted ones and stealing the identity.

The uPort system also consists of a registry contract which stores encoded links to users' data as well as information contained in users' public profiles. A user can specify the information to make public and still have control over their data. The data is stored in a distributed open-source database known as the Interplanetary File System (IPFS) and not on the Ethereum blockchain or the uPort smart contracts. Only encrypted links to identify the data are stored in the blockchain and can be used to verify that attestations are valid. Data can be attributes and attestations of the identity that take the form of a signed JSON web token. This type of digital signature is a way for other users e.g. financial service providers to verify facts about an identity. Other possible data include public key certificates and links to social network profiles.

uPort specifies developer libraries which enables service providers to easily combine its system into their own applications in order to interact with users through their uPort identities. Some operations include logging with QR codes into uPort, signing transactions and querying data.

The advantages of uPort apart from the benefits of blockchain (Dunphy and Petitcolas, 2018) are:

- A user has more control of their attributes and attestations and who they may share it with.
- A user's address does not supply information about a particular user, unlike, for example, a South African identity number which begins with a citizen's date of birth.
- Use of the same mobile application across different service providers allows there to be consistency in the system which should improve user experience.
- A user may create multiple identities for different interactions which means that the registry does not need to be queried so often.
- Its design on the Ethereum platform allows identities to engage with other non-uPort contracts regarding assets etc.

Possible risk areas mentioned by Dunphy and Petitcolas (2018) include:

- Users are more responsible for their identities and those identities for which they are recovery delegates.
- The registry contract allows a central storage space which can be susceptible to attacks.
- A node can recognize that a smart contract at a certain address represents a uPort identity which can compromise user privacy.
- A user is unlikely to store negative attributes, for e.g. poor credit score, which is good for the user, but not good for service providers that need to verify whether an identity is reliable.

Furthermore, there is a lack of a governance model to ensure that verifiers can be trusted and the reliance on Ethereum means that it inherits any issues with the platform.

Note that the concept of a controller contract for each user, as described in the whitepaper by Lundkvist et al. (2016) and explained here, was considered in the initial release of uPort. Since then their system has improved and instead of each user having a controller contract, there is a single "Identity Manager" contract that performs the same operations as the controller contract for all identities and interacts with any proxy contract as long as it is the correct owner (uPort, 2018).

2.2.2 Sovrin

Sovrin (Tobin and Reed, 2017; Sovrin Foundation, 2018a) is a self-sovereign identity system with its own blockchain ledger to record identity transactions. It was started by the Evernym Corporation and evolved to be managed by a global, diverse and non-profit group called the Sovrin Foundation. The blockchain can be accessed and read by the public, however, only reliable organizations ("stewards") accepted by the Sovrin Foundation can write and validate transactions i.e. it is permissioned. The Foundation also monitors the ledger and its open source code based on a set of legal and technical policies. This review is also based on documentation by Windley (2016) and Tobin (2017) that describe the workings of the Sovrin system.

A user interacts with the Sovrin network through applications supplied by other providers and which enables one to manage and share their identity information securely. Sovrin does not appear to offer their own application and their focus is on designing the infrastructure to securely support the recording of claims and other transactions. Sovrin does not specify smart contracts, however, a smart contract platform could be used by an agent providing the application. For example, Tykn (2018) offers an application for identity registration and verification services that combines the Sovrin identity platform and the Rootstock smart contract platform.

Unlike uPort (Lundkvist et al., 2016), the Sovrin system requires different identifiers for different relationships which further helps to keep a user's data private and from being shared among third parties without the user's consent. As Windley (2016) explains, identifiers are a

pair of cryptographic keys, a public key to verify claims and a private key to sign claims. It is based on a specific signature scheme known as Ed25519. The system also has a JSON type document ("DID document") that includes the public key and network addresses of the parties in the relationship (Tobin, 2017). Claims are statements about the user made by the user itself or issued by another party. When a claim is issued by a trusted entity and signed with their private key, then it is a verifiable claim or proof and the user can use the provider's public key to confirm that the source is that provider. A major service provider does not need to have a separate key-pair for each client relationship and can have a public key that is known to many users. An individual shares only their public key with a service provider and only for that party, who can use it to verify that any information that their client shares with them originates with that client. The public keys of both the individual and the provider are recorded in the ledger, while private keys are stored by each entity on their own devices. The provider can also confirm that a verifier's signature on a claim is valid by using the verifier's public key.

A private ledger can be created for a user and managed by an agency that must comply with the policies of the Sovrin Foundation. This ledger can store the user's identifiers and claims, and a hash of this data can be recorded on the public ledger as a form of public proof of identity without sharing the underlying data. Windley (2016) also explains that there are different types of claims that can be defined with a certain structure and this definition can be recorded in the ledger and referred by claims so that users can understand its contents. Claims can be revoked or have a time limit in which they are valid. A consent receipt is a special type of claim that defines conditions on the use of data in a claim, for example, an expiry date. Claims can be made public, such as the location of an organization, or private. Public claims can be stored on the Sovrin ledger and verified by any user. This increases trust in certain social or public services. Private claims and proofs from others are not contained on the Sovrin ledger but can be stored in users' private ledgers.

Disclosure proofs are a way for a user to create a new verified claim containing several full claims or partial claims. It allows claims to be combined and re-used and can contain specific information about the claims for the user to control what details to share. Another special cryptographic key known as a master secret is attached to it. A shared party can verify all the containing claims. The identifiers for these claims are not linked in a way that an identifier at one issuer can be related to an identifier at another issuer. Thus, the privacy of data is not compromised.

A user's identity is their collection of identity transactions and relationships with other parties. Its contents are not contained in a specific location but are distributed in the main ledger and the user's private ledger. The Sovrin network also includes a token to exchange value between users (Sovrin Foundation, 2018a). There is an opportunity to create a marketplace where verified claims can be issued at a price. This encourages participation and good behaviour in the network.

The benefits of Sovrin include:

- Compared to uPort, there is more privacy with the use of identifiers for every relationship and private ledgers for identity information.
- Since not all users can write to the network, there is less risk of a group of untrusted entities combining their computational resources and gaining control of the network (Tapscott and Tapscott, 2016).
- The ledger only stores transactions related to identity.
- A way for businesses to generate revenue from issuing verified claims.

Some disadvantages of Sovrin are:

- There is no universal application to access the network and users need to rely on other agents to provide that service.
- There needs to be multiple stewards from a range of countries to ensure that the network does not become centralized (Grüner et al., 2018).

2.2.3 Civic

Civic (Lingham and Smith, 2018) is a system that uses the Rootstock smart contract platform on the Bitcoin blockchain to help facilitate identity verification. There already exists a market for such services, but these processes take up time and have high costs. With the use of blockchain and smart contract technology, Civic aims to provide a faster, cheaper, more accessible and secure method. Lingham and Smith (2018) explain how consumers even in a developed country such as America can have limited access to services due to a lack of verified financial history and other identity-related information. Furthermore, there are regulations that restrict the sharing of personal information without the consent of the owner. Thus, with the current identity systems, different service providers cannot simply share the identity information of a customer with each other. The Know Your Customer process has to be repeated across businesses.

A user interacts with the Civic system from a mobile application. They may submit identity information to Civic, which currently performs the work on the user's behalf of finding a verifier and obtaining a verification of the user's data. Civic confirms the validity of the information and signs that it is verified by them or an identity partner. The proof of this transaction along with an encoded hash of the data is recorded on the blockchain. The verified data is sent back to the user and none of the user's private information is retained by Civic or the verifier. Data is stored securely with the use of encryption and biometrics on the user's device. When a service provider requests credentials from a user, the user may send them verified information and the requestor is able to check that this attestation is authentic with the record on the blockchain. Organizations can use QR codes to request information from a user who scans the code, checks what data is needed and can agree to send it.

Lingham and Smith (2018) explain that the current Civic system is being improved so that other reliable identity verification providers can be included in their network and validate user claims. When a user applies for a service or product, they can re-use their Civic credentials without sharing the underlying information, thereby not compromising any privacy regulations. The new system functions with smart contracts and consists of a CVC token that can be used for rewards and payments. A verifier can charge other service providers for re-using the credentials they sign. This is expected to be cheaper than the current cost of verification. The token also incentivizes Civic members to participate in this network and remain trustworthy. Smart contracts will contain the details of validators and the price of existing or new attestations which can be updated at any time.

The advantages of Civic apart from user control are:

- The use of a well-established Bitcoin network.
- A universal application with biometric functionality which is convenient for users.
- A marketplace for verification which is attractive to businesses.

Possible risks include:

- The difficulty in ensuring that validators can always be trusted.
- The reliance on a public ledger with the threat of untrusted nodes combining their electronic resources and gaining control of the network (Tapscott and Tapscott, 2016).

2.3 Funeral insurance

Funeral insurance is a financial product that assists the holder of the policy or his beneficiaries with funeral and related expenses (see, for example, Old Mutual (2018)). The insurer collects a certain amount (called a premium) every month from the policy holder and, in the case of death, pays the value of the cover to the holder's beneficiary. For regulatory purposes (Financial Intelligence Centre Act, 2001), the applicant needs to show a valid identity document and recent proof of residential address to the insurer in order to obtain this product. Based on the products offered by several insurers in South Africa (Old Mutual, 2018; Liberty, 2018; Sanlam, 2018), there are different types of funeral plans that provide cover for just an individual, or immediate family, or even extended family. The main benefit is usually up to 100 000 rands. For larger covers, the applicant would also need to supply medical information. Generally, funeral policies include additional benefits to help cover airtime, grocery and education expenses for the remaining family, as well as counselling options and legal advice on wills. There can also be conditions such as a waiting period, payment holiday and premium payback. A waiting period is an initial amount of time that must pass before an insurance claim can be made. A payment holiday specifies the number of payments a holder can miss and still remain covered. A premium payback refers to premiums that are paid back to the holder if no claim has been made over a long period of time.

3. Development of a self-sovereign identity system

3.1 Use case for funeral insurance

In this section we develop an application of self-sovereign identity that is beneficial to both the individual and business. There are a number of use cases such as (i) digital government services (e.g. Zug uPort identity (Offerman, 2018)), (ii) academic records (Bertram and Georg, 2018), (iii) Civic's proposed marketplace for the use of credentials (Lingham and Smith, 2018), (iv) Bitnation's digitally-based nations (Tarkowski Tempelhof et al., 2017). These range from the case of translating existing physical services to digital ones, to the case of creating new organizations and economies. We focus on the re-usability of self-sovereign identity in a financial corporation that sells multiple funeral insurance products. In such companies there can be similar products offered by different business units that service different markets (Birch, 2007). Suppose that in this corporation there is a Youth Markets Division and Family Markets Division that sell products suited for young individuals and families, respectively. We consider the case where a customer purchases two funeral policies at different times from these different departments. We investigate how a SSI system can improve on the existing process so that identity verification re-work and storage of related information are avoided, which saves on time and cost for the insurer and increases privacy for the customer. Information on the first policy obtained from Youth Markets can also be shared with Family Markets to determine if a client is over-insured. In terms of Treating Customers Fairly regulations (The Banking Association South Africa, 2018), this concept is important.

The motivation for this use case is twofold. Firstly, funeral insurance is a popular financial product in South Africa with residents placing an importance on a funeral as a way to honour the life of the deceased (Roth, 2000). It is a cultural expectation among many South Africans to hold a funeral and cater for many attendees. Thus, the cost of this event can become quite high (Case et al., 2013). As a result, there is a lucrative business for selling funeral policies to cover funeral expenses and many organizations offer this product (for e.g. Old Mutual, 2018; Liberty, 2018; Sanlam, 2018). Insurers include the cost of operating their business in pricing their products (Cummins, 1991). With self-sovereign identity management, the cost of re-verifying customers and storing their data securely will be reduced and insurers can offer funeral products at cheaper prices which will be a significant benefit to their customers. Secondly, there are expectations of new technologies having the potential to disrupt centralized businesses or even replacing them (Skan et al., 2015; Sironi, 2016). There are also many attempts to create new types of services. However, it would require time for new models to be accepted especially since they would first need to be regulated to ensure business accountability and customer fairness (Tapscott and Tapscott, 2016). This motivates current businesses to consider how new technology can be used to improve their processes and business value. Moreover, businesses should aim to remain relevant as prospective

younger customers are more likely to accept innovative solutions (Papp and Matulich, 2011) and businesses would want to appear more attractive to them.

3.2 System design

Our use case consists of the below entities:

- Insurer – A financial service provider that sells two funeral insurance products, a Single Funeral Plan and a Family Funeral Plan, which are offered by different divisions, Youth Markets and Family Markets, respectively. The Single Funeral Plan covers only the holder while the Family Funeral Plan also allows relatives to be covered. These plans may include benefits for sundries and legal advice and conditions for the waiting period, payment holiday and premium paybacks.
- Customer – An individual who has documents to prove his South African identity and residential address. He will apply at different times for the above two products from the insurer. He is a new customer to the business.
- Customer's relative – An individual with a valid proof of identity. He will be covered as a dependent under the customer's Family Funeral Plan.
- South African Government – An organization that verifies an individual's identification document and proof of residential address.

Suppose that the current process works as follows. The customer contacts the insurer in-person at a branch or by online or telephonic means to apply for a Single Funeral Plan from Youth Markets. A corporate employee accesses his business unit's application. Since the client is new, he creates a profile for the client with the client's identity number. Although a business may create their own customer identifier, it should link to the customer's identity and an identity number would be used since it is a unique identifier. The customer supplies a copy of his identity document and a recent proof of his residential address, such as a municipal utility bill, as verification of his identity (Financial Intelligence Centre Act, 2001). The employee checks that the customer is a South African citizen and is older than 18 to be eligible for funeral cover. Depending on the customer's financial needs, a Single Funeral Plan contract is created for the customer with a suitable premium, insurance value, benefits and conditions. Youth Markets stores the customer's product information as well as the customer's detail in their database.

At a later time, when the customer applies for a Family Funeral Plan from Family Markets, this process is repeated. Another profile is created for the client since he is new to that division. For regulatory purposes, the customer's personal information is again collected and stored in Family Markets's database. The customer adds his relative to be covered by his plan and the insurer requests a copy of the relative's identity document which is also retained by Family Markets. A Family Funeral Plan contract is then generated for the customer. This process is time-consuming for both the customer and insurer, and there is a clear inefficiency in storing duplicated data and not sharing data between departments.

In summary, the actions taken in this process with either department are:

- Customer applies for funeral cover from insurer.
- Insurer adds customer to their records.
- Insurer requests information from customer.
- Customer shares information with insurer.
- Insurer issues customer with a funeral plan.
- Insurer saves product and customer information.

As depicted in the below diagram, information flows between the customer and insurer through applications on the corporate's servers and stored in databases also on their servers.

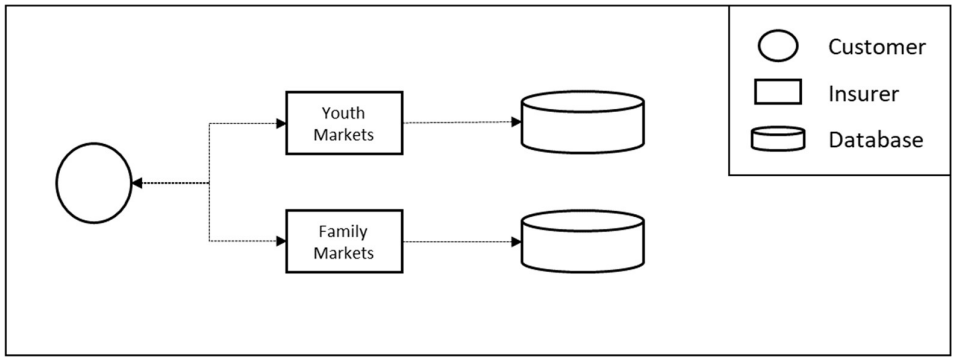


Figure 1. In the current process for a customer that applies for two funeral plans at different times from different departments (Youth Markets and Family Markets) within an insurance company, identity information flows from the customer to both departments and stored in their databases.

We design a self-sovereign identity system that consists of a decentralized application on the Ethereum smart contract and blockchain network. We note the following definitions, keeping the same terminology as used in the literature (Windley, 2016; Lundkvist et al., 2016):

- Claim – A statement about an entity e.g. "Is a South African citizen". This may be created by the entity itself or issued to the entity by another party.
- Verification or verified claim – A claim that has been verified as being true by a reliable authority e.g. the South African government.
- Identity – An entity with various claims, verified or unverified.

The customer, relative, insurer and government each have smart contracts which represent their identities and which they use to interact with each other and store any encrypted claims. There is a main factory contract that creates these identities by deploying instances of their respective contracts to the network. The blockchain records the creation of the contracts and any changes made to it in a ledger. Copies of the ledger are distributed across all nodes in the network. There is one application that can be used by all entities to access the network and it

is not located on the insurer's server, but contained and run on the user's device. Each entity also needs a wallet account such as a MetaMask account (ConsenSys, 2018b) to connect to the network and run the application. The insurer only stores the customer's product information in their database, while the customer's identity and product information is saved on his own device. The components of this system are explained in more detail in the next section. The below figure illustrates this system.

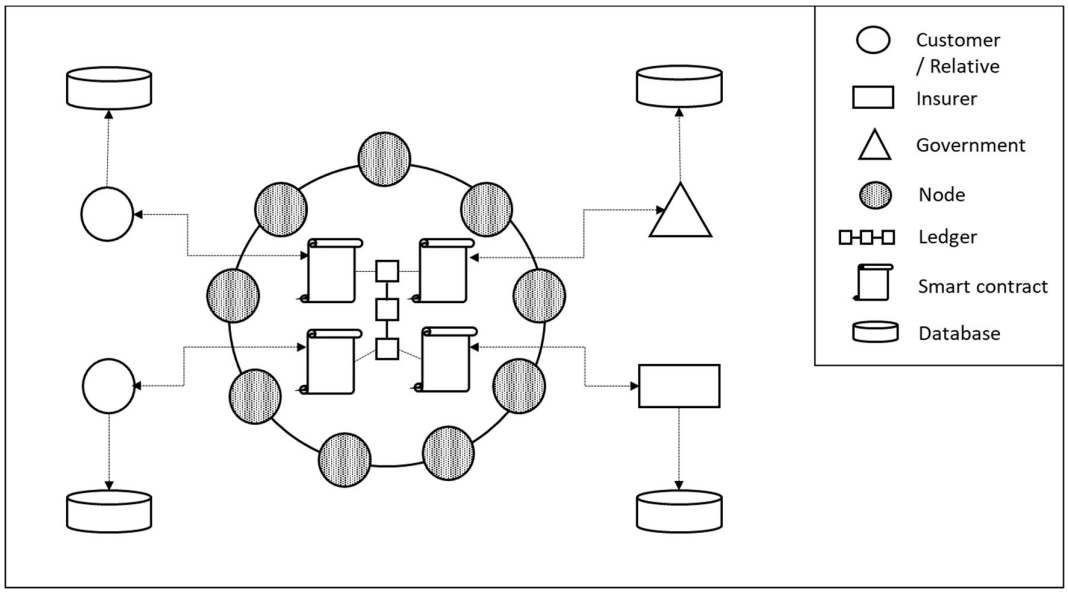


Figure 2. In a self-sovereign identity system for funeral insurance, the entities interact with each other through their smart contracts which are contained in a decentralized ledger managed by a public network of nodes. The customer and his relative obtain verification of their identity claims from government and share them with the insurer. The insurer issues verified claims on funeral plans to the customer.

Each entity manages their identity through their wallet, which employs public-key cryptography and is a common means to connect to decentralized networks (Lundkvist et al., 2016). A wallet contains two alphanumeric keys that are cryptographically related but such that it is difficult to calculate the one key from the other. One key is private so that only the owner is aware of it, while the other key is public and can be shared with other entities. Assuming the user keeps their private key safe, their connection to the network is reasonably secure. The separation of the private and public elements for the customer is shown in the below figure. The other entities are similar. Note that the insurer only has one identity in the system and both Youth and Family Markets share the wallet credentials. In practice, both departments should also have their own system to manage access to the wallet by employees.

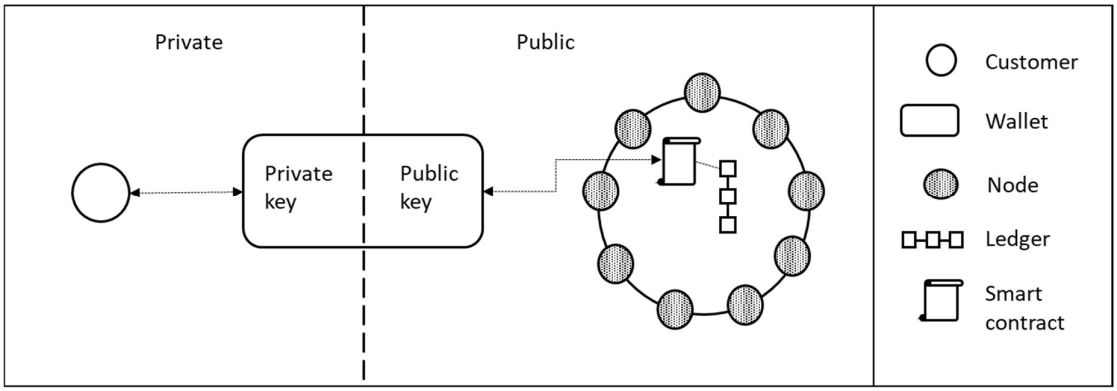


Figure 3. Separation of private and public elements for a customer interacting through a wallet with a self-sovereign identity system for funeral insurance.

In this system each user can be uniquely identified by a public address on Ethereum. This address is twenty bytes in size and is a cryptographic hash of the user's public key from their wallet account (Wood, 2014). Unlike a South African identity number which includes one's date of birth, the Ethereum identifier does not provide any personal information about the user. An instance of a contract also has an address that is used to identify the contract.

The contracts have the following features:

- Apart from creating identities, the factory contract also stores public claims from the government and the insurer on the verification service and funeral plans that they respectively offer, and these can be viewed in the application.
- The customer's contract has options to create claims on his identity, residential address and age, and to request verification on these claims from the government. The information used to create a claim is encrypted and only its identifier and not the supporting data is stored in the contract. The customer can also share a claim with the insurer when requested and receive claims issued to him by the insurer.
- The customer's relative's contract has the same functionality as that of the customer.
- The government can use their contract to verify claims from the customer and his relative.
- The insurer's contract has options to add a customer, add a dependent for a customer, and store claims that the customer shares with the business. The insurer can also issue claims that the customer has a Single or Family Funeral Plan.

The application is web-based and has functionality to connect an entity to their smart contract and allow execution of its operations. The customer, government and insurer can also view their claims, verification requests and shared claims, respectively, in the application. The supporting data used to create claims is formatted in JSON and saved as a text file from the application onto the customer or insurer's device.

With this SSI model, the use case works as follows. The customer logs onto the application with his wallet account and sees that the insurer offers a Single Funeral Plan with a contact number for the Youth Markets Division. He contacts the department to apply for the product. Since he is a new client, he signs up as a customer on the application. This action submits his public address to the insurer. A Youth Markets employee requests proofs of the customer's identity, residential address and that he is older than eighteen. The customer shares his claims on this information and the employee checks that these claims are verified by government. The employee also checks the expiry date of the address claim to ensure that the claim is still valid. When a Single Funeral Plan that suits the customer's needs has been created, Youth Markets issues the customer with a claim on this product. The data underlying the product claim is saved to their database and also sent to the customer privately.

At a later time, the customer applies for a Family Funeral Plan from Family Markets and submits his relative's identifier as a dependent. The department sees that he already has verified identity information and can proceed. A Family Markets employee requests proof of identity from the dependent and the relative provides his verified claim. The insurer also observes that the customer has a claim for a Single Funeral Plan and may request the claim data from the customer. This data is shared between the entities separately from the blockchain system (referred to as off-chain). The business checks the insured value of the customer's first funeral plan to ensure that the he is not overinsured with the new policy. Family Markets then issues a claim to the customer that he has a Family Funeral Plan and sends him the supporting data off-chain. The product data is also saved by the business unit. This SSI method allows the insurer to save on time and storage space as client credentials can be re-used and sensitive information does not need to be stored. It also offers the customer a better user experience. Since the request and receipt of claims are recorded on the blockchain, it is available for auditors to check that the insurer complies with regulations.

3.3 System components

We design an interface that each entity can view in a web browser and use to interact with the rest of the system. This interface is built using HTML to display the content, CSS and Bootstrap to provide style for the content and JavaScript for user interactions. The application can be downloaded to the user's computer and when it is run, it connects to the decentralized Ethereum blockchain rather than a centralized server.

Ethereum (Wood, 2014; Buterin, 2014) is a blockchain that enables code to be stored on its network and executed in transactions which are then added to the ledger during the mining process. This code is contained in objects called smart contracts. Unlike the Bitcoin blockchain that can only perform a certain set of instructions, i.e. transferring Bitcoin, Ethereum allows for different types of programs to be run automatically. This is carried out by a special program called the Ethereum Virtual Machine (EVM). Solidity (Ethereum, 2018) is the language used to write the smart contracts and a compiler converts it into a language that is understood by the EVM (Modi, 2018). The network is public so any user can read and

write information to the ledger. Since writing to a blockchain requires a node to use expensive computing resources and electricity, the miners are paid for this in the form of a digital currency called ether. Gas is another currency defined on this platform and it is used to fix the cost of operations (Modi, 2018). The exchange rate from gas to ether is referred to as the gas price and this price changes so that the amount of ether needed to run an operation is reasonable. This is necessary due to the instability of the ether currency relative to the dollar (Corbet et al., 2018).

Our SSI system consists of a main factory contract that creates other contracts for each entity. These smart contracts are a way to transfer identity verifications between entities and their recording on the blockchain is evidence that it occurred. We demonstrate the system using Ganache (ConsenSys, 2018a) which provides an Ethereum node and a ledger that runs locally on a computer for development purposes. It provides ten accounts each with 100 ether, a public address and a private key that limits access of the account to just the owner. The public addresses allow one to send and receive transactions. We use four of these accounts to represent the entities in our system. We also use the Truffle framework (ConsenSys, 2018c) that enables one to compile, deploy and test smart contracts (Bertram, 2018). It provides a certain project structure that makes development more convenient and a truffle-contract library to work with the contracts in the application.

When a user accesses the application, a wallet account provided by a third party is required to connect to the smart contracts from the application and to pay in ether for operations to be executed (Modi, 2018). We use a MetaMask wallet (ConsenSys, 2018b) which is an extension within a browser. We add four Ganache accounts for the entities to the wallet. Other tools that are required to connect our application to the network include the JavaScript libraries Node.js (Node.js Foundation, 2018) and web3.js (Ethereum, 2016).

Claims on identity and funeral plans are created with data supplied in the application. There are functions in the smart contracts to combine and hash data into an encrypted identifier which is stored with other claim information in the contracts. The application aggregates the data and its identifier into a JSON object and outputs it to the user's device in text form. The user is responsible for managing their data and other software can be used to store the data in a local database for the user. The customer owns their data completely and may choose to share their verified claims with other parties without sharing their actual data.

The application can be found at https://github.com/JoMoodley/Identity with instructions for how to deploy the contracts and run the system.

3.4 Smart contracts

In this section we describe how the smart contracts are coded in Solidity. The Solidity concepts that we refer to can be found in material by Ethereum (2018) and Modi (2018). The source code is provided at https://github.com/JoMoodley/Identity/contracts.

3.4.1 General concepts

A contract is an object that contains state variables to store information, functions to conduct operations and events to record actions which can be tracked by the front-end application. Our SSI system has a main factory contract called `IdentityFactory` and three entity contracts `IdentityGovernment`, `IdentityInsurer` and `IdentityCustomer`, where the last contract is used to create instances for both the customer and his relative. Most of the variables we define are private which means that they can only be accessed and modified from functions within the contract itself. The factory contract contains some public variables which can be viewed by any user external to the contract, however, they can only be updated from operations inside the contract. Similarly, functions that are declared public can be called from outside the contract, but private functions can only be executed from within the contract. Some functions are defined as viewable which means that they read contents from the contract, but do not modify any of the information.

The system also includes contract interfaces that enable the identities to interact with each other. An interface is a version of a contract that only contains the definitions of the functions that need to be called from other contracts. For example, when the customer's contract needs to interact with the insurer's contract, it refers to an insurer contract interface at the insurer's contract address. It is then able to access the relevant functions in the insurer's contract, provided those functions are specified in the interface.

The msg.sender variable is a special element in Solidity that represents the address of the caller of a contract. When a user directly accesses a function, msg.sender is their account address. However, if the customer contract, for example, calls the insurer contract, the msg.sender is then the customer's contract address.

It costs gas to deploy a contract and to execute functions that change information stored in the contracts or create new information. In Ganache there is a maximum amount of gas that can be used to deploy a contract. Thus, in designing these contracts, the maximum cost of deployment is considered. Where possible, we avoid duplicating operations by defining private functions that can be called from multiple other functions. There is a memory limit in Solidity for the number of variables including inputs that a function can use. Thus, for functions with too many local variables, we also use private functions to perform subsets of operations. A string variable is typically used to store a statement of any number of characters, however it increases the size of the contract significantly, so we use a bytes32 data type instead to store a character statement.

We use Solidity version pragma ^0.4.24 which works for compiler versions 0.4.24 up to and excluding 0.5.0.

3.4.2 Factory contract

The main purpose of the `IdentityFactory` contract is to create identities, thus it is only necessary to deploy this contract. It imports the code of the other contracts so that it can deploy instances of those contracts.

In this contract we define a structure `ServiceClaim` to describe a claim about a service or product that the government or insurer offers. A structure is a customized set of variables. This structure contains a variable to store the provider's public address, two bytes32 variables to specify the name of the provider and a description of the claim, and an integer variable for the contact number of the provider. The contract contains the following state variables:

- A private variable that stores the government's public address which is used to identify the government.
- A private variable that stores the insurer's public address which is used to identify the insurer.
- A public integer to track the number of identities that are created. It is initialized to zero.
- A public array of `ServiceClaim` structures to store the providers' claims on their services and products that can be viewed by any user.
- A private variable `ownerToContract` to map an entity's wallet account address to their contract address. When a contract is created, it is deployed to a specific address on the network that is different from the user's account address.
- A public mapping variable `ownerToIDType` that links the user's account address to an integer indicating the type of identity. The types 1, 2, 3 and 4 represent the government, insurer, customer and relative, respectively.

We define an event `NewIdentity` which is called when an identity has been created. The front-end application watches for events and updates its content when the event has occurred. There is a public function `createIdentity` to create identities for each entity and it requires an integer input for the type of identity. When the function is called by a user, the msg.sender variable is the user's wallet account address. First the function checks that the entity does not already have an identity by requiring that the mapping of the sender's address to their contract address is zero. Then different operations are performed for the different types of identities. A contract for the insurer is created with the statement

```
IdentityInsurer corp = new IdentityInsurer(msg.sender),
```

where the insurer's account address is submitted as an input into the contract. The `ownerToContract` mapping of the sender's address is set to the address of the contract. The contracts for other entities are created similarly. Once an identity has been created, the sender's `ownerToIDType` mapping is set to the relevant identity type, the identity count is incremented by one, and an event is emitted. There are also checks to ensure that only one

government and one insurer is created by confirming that their address variables stored in the contract are zero. Once their identity is created, their address variable is set to their account address. For the government, a service claim is created with their account address, name as "SA Government", description as "Verifier" and a contact number (a fake one is used for demonstration purposes). This claim is appended to the array of service claims. Similarly, two claims are created for the insurer with the name "Insurer" and descriptions "Single Funeral Plan" and "Family Funeral Plan".

There are three public viewable functions:

- `getServiceCount` outputs the number of service claims using the length attribute of an array.
- `getGovAddress` returns the public address of government.
- `getContractAddress` provides the contract address for an entity. To obtain the government or insurer contract address, the identity type 1 or 2 must be specified. To obtain the customer or relative contract address, their account address needs to be supplied.

3.4.3 Entity contracts

The entity contracts have the following common features:

- An interface is defined for the contract to interact with the factory contract and specifies the `getContractAddress` function. The customer's factory interface also includes the `getGovAddress` function, since the government's account address must be set as a verifier of the customer's personal claims.
- There are state variables to store the address of the owner of the contract and a reference to the factory contract. The owner's address is a private variable.
- A constructor is a function that is only executed once when the contract is created. It sets the owner of the contract to the entity's account address. Since the entity contract is deployed from the factory, the msg.sender of the constructor is the factory contract address. The factory interface at that address specifies the reference to the factory contract.
- A modifier is a special function that can specify restrictions on other functions. A modifier `onlyOwner` is defined that checks that the caller of a function is the owner of the contract. This ensures that some functions can only be accessed by the relevant entity.

In the `IdentityCustomer` and `IdentityInsurer` contracts we define a structure `Claim` for a customer's claim information. It contains a description of the claim, an alphanumeric identifier of the data supporting a claim, an integer value for the expiry date, a boolean value specifying whether the claim is verified or not, and the name and address of the verifier. The claim description, data identifier and verifier's name are stored as bytes32 variables. In the

`IdentityGovernment` contract a structure `VerificationRequest` is defined to store a request to verify a claim. It consists of the above claim information as well as the requestor's address and claim number. The `IdentityInsurer` contract also describes a structure `Customer` to store a customer's address, the address of their dependent and an array of claim structures that the customer shares with the insurer.

The customer contract stores a private array of claim structures as a state variable. There are two public viewable functions `getClaimCount` and `getClaim` that can be used by only the customer to retrieve the number of claims and a claim at a given index in the claims array, respectively. The customer and insurer contracts also have an integer variable `claimNonce` that is used in the conversion of claim data into an encrypted identifier and an event `NewClaim` that is emitted with the data identifier when a claim is created. The government contract stores a private array of verification requests from users. As a claim is verified, it is removed from this array. A function `getVerificationRequestCount` provides the number of requests currently in the array and a function `getVerifRequest` returns a specific request from the array. These functions are public, viewable and have the `onlyOwner` modifier so that they can only be accessed by the government.

The insurer contract stores a private array of customer structures and a private mapping `customerIdToNum` of a user's account address to their customer number. The mapping is used to locate a customer in the customer array given their address. The contract contains functions to output the number of customers and the number of claims shared by a customer. The insurer can also use their `getCustomer` and `getCustomerClaim` functions to obtain the identifier, dependent's address and shared claims for a particular customer in the array. These functions are all public, viewable and only accessible by the insurer. The `addCustomer` function is executed when a customer signs up to the insurer in the application. It first checks that the customer does not already exist by requiring that the mapping of the caller's address to a customer number is zero. Then it appends the customer's identifier to the customer array. There is also a function `addDependent` that can be called by the customer to add an address for his dependent.

The entity contracts contain operations to create, verify, share and receive claims and these will be discussed in the next sections.

3.4.4 Creating and verifying claims

Here we explain how a customer creates a claim and obtains a verification from the government. We define an interface `GovInterface` for the customer's contract to interact with a function in the government contract to add a request to verify a claim. We also specify a contract interface `CustInterfaceG` which is used by the government contract to add a verification of a claim in the customer's contract.

The `IdentityCustomer` contract has three public functions for a customer to create claims for their proof of South African identity, proof of residential address and proof of age. Each function can only be called by the owner of the contract and requires a set of data attributes as described below:

- `createClaimID` function – Inputs integer values for the customer's identity number issued by government and their date of birth, and a character variable for the customer's name.
- `createClaimAddress` function – Inputs character variables for the address line 1, address line 2, town and country, and integer values for the postal code and expiry date of the claim.
- `createClaimAge` function – Inputs an integer value for the customer's date of birth.

In the `createClaimID` function, the claim data is combined along with the entity's account address and nonce and then converted into an encrypted data identifier using the keccak256 hash function in Solidity. This action is carried out with the following statement:

```
bytes32 dataId = keccak256(abi.encodePacked(_idnumber, _name, _dob,
                        owner, claimNonce)).
```

The identifier is an hexadecimal string of size 256 bits. The encryption method ensures that it is difficult to obtain the data given the identifier and that similar data inputs produce different results. The nonce is increased by one when any claim is created so that it is used uniquely. If a customer creates a second claim on his identity using the same data, a different nonce would be used and the resulting data identifier would be quite different from the first identifier.

The function then calls a private function `_addClaim` that takes in the data identifier, claim description "South African identity" and expiry date 0, and uses these attributes to create a claim structure. In this subfunction we obtain the government's account address from the factory contract and set the verifier of the claim to that address. The verifier name is set to "SA Government" and verified status is false. The claim is appended to the customer's claims array. The main function then calls another private function `_requestVerification` that takes in the length of the claims array as the current claim number. In this subfunction we obtain the government's contract address from the factory contract and using the customer contract's government interface, we create a local reference to the government contract at their address. The customer contract then calls the `addVerificationRequest` function in the government contract, supplying the customer's address, claim number and claim information. Finally, the `NewClaim` event is emitted with the data identifier which is used in the application to create a data file for the user. Claims for address and age are created similarly with the descriptions "Residential address" and "Age over 18 years", respectively. For the address claim, the expiry date specified by the customer is used.

In the government contract, the `addVerificationRequest` function creates a verification request structure using the requestor's address and claim details and appends it to the array of requests. It requires that the claim verifier is the government's account address and that the caller is the contract address of the given requestor. There is a function `verifyClaim` that can only be invoked by the government to verify a particular claim. It obtains the customer's contract address from the factory contract and using the government contract's customer interface, it creates a local reference to the customer's contract. It then calls the `addVerification` function in the customer's contract providing the relevant claim number. The claim request is removed from the government's requests array by shifting all the subsequent requests one position down. In the customer's contract, the `addVerification` function first ensures that the caller is the government's contract address and then sets the verified status to true for the relevant claim.

3.4.5 Sharing and receiving claims

Here we explain how a customer shares a claim with the insurer and how the insurer issues a claim for proof of funeral cover to the customer. We define the interfaces `CorpInterface` and `CustInterface` for the customer contract to interact with insurer contract and for the insurer contract to interact with the customer contract, respectively.

The `IdentityCustomer` contract contains a public function `shareClaim` that is only accessible to the customer. It obtains the insurer's contract address from the factory contract and using its interface with the insurer, it creates a local reference to the insurer's contract. It then calls the function `addSharedClaim` in the insurer contract, and sends its address and the claim it has selected to share with the insurer. The insurer contract checks that the function was invoked by the sharer's contract address. Then it retrieves the customer number corresponding to the sharer's account address so that it can identify the position of the customer in the customers array. It creates a claim structure with the inputted attributes and appends the claim to the shared claims array for the relevant customer.

The `IdentityInsurer` contract includes a `createFuneralPlan` function which only the insurer can use to create a claim that the customer has a funeral plan. It requires the following inputs:

- Type of plan - 1 for Single Funeral Plan or 2 for Family Funeral Plan.
- Integer values for the customer number, premium, amount of cover, amount of sundries benefit and the waiting period in days.
- Character variables for conditions on the payment holiday, premium paybacks and whether legal advice is offered.

These attributes, excluding the plan type `and` customer number, are combined with a nonce and hashed to produce a unique data identifier which is added to a data file in the application using the `NewClaim` event. A private function `_addClaim` is called that takes in the

customer number, data identifier and description “Single Funeral Plan” or “Family Funeral Plan” to create a claim and append it to the relevant customer’s shared claims. It sets the claim expiry date to zero, the verifier name to “Insurer”, the verifier as the insurer’s account address and the verified status as true. It then uses the customer’s contract address from the factory and its interface with the customer to issue the verified claim to the customer. The function `receiveClaim` in the customer’s contract is invoked, which checks that the caller is the insurer’s contract address before adding the claim to the customer’s array of claims.

3.5 Application

Here we describe how the front-end application is designed for a user to view information and carry out operations contained in the smart contracts. Since the code is similar across most of the application methods, we only explain the code in some cases. We use the truffle-contract (ConsenSys, 2018c) and web3.js (Ethereum, 2016) libraries, and examples provided by Bertram (2018) to develop the application. Refer to https://github.com/JoMoodley/Identity/src for the source code and https://github.com/JoMoodley/Identity/blob/master/README.md for instructions on running the system.

The HTML user interface is separated into sections to show different content for the customer, government and insurer identity types. There are also sections for the below functionality:

- Set up the system by creating identities for the government, insurer, customer and his relative.
- Display the services offered by the providers.
- Display the identity type and public account address of the user.

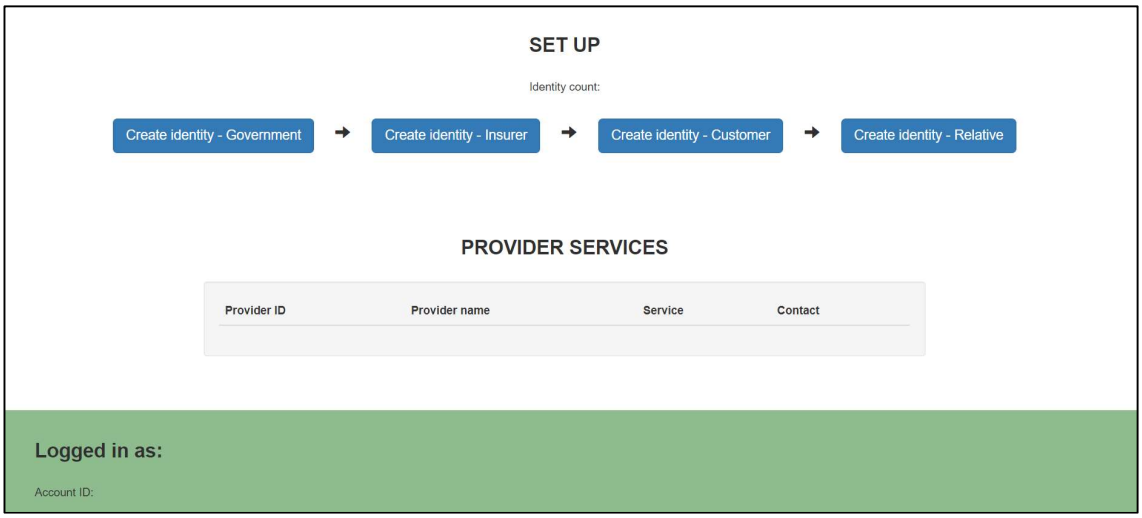


Figure 4. SSI application – Sections to create identities and display provider services, user's identity type and user's public account address.

In the customer section, the customer or relative can view their own claims and create claims for their identity, address and age by entering data in a form and submitting it. They can also sign up to the insurer, add a dependent's address to their contract and select a claim to share with the insurer. The government can view requests added to their contract and select a request to verify. In the insurer section, a list of their customers is shown and the insurer can select a customer to view their shared claims. There are also forms to enter data for a Single or Family Funeral Plan and submit it to create a claim for the customer. Below are illustrations of these sections.

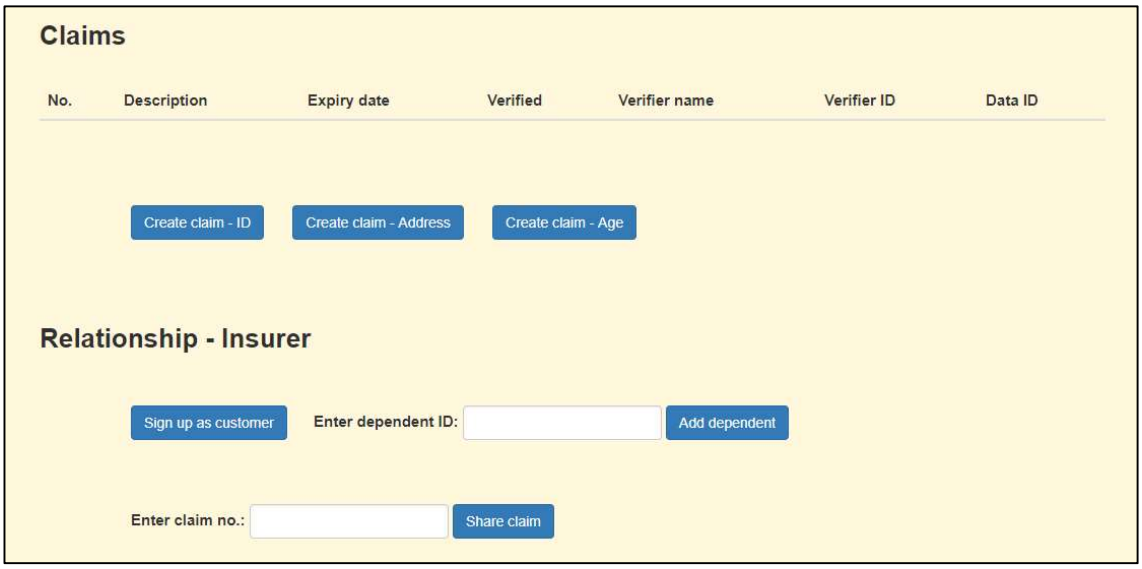


Figure 5. SSI application – Customer section.

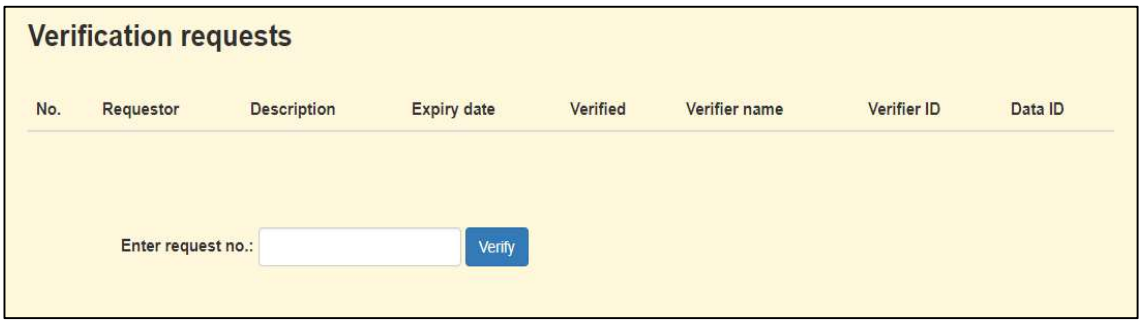


Figure 6. SSI application – Government section.

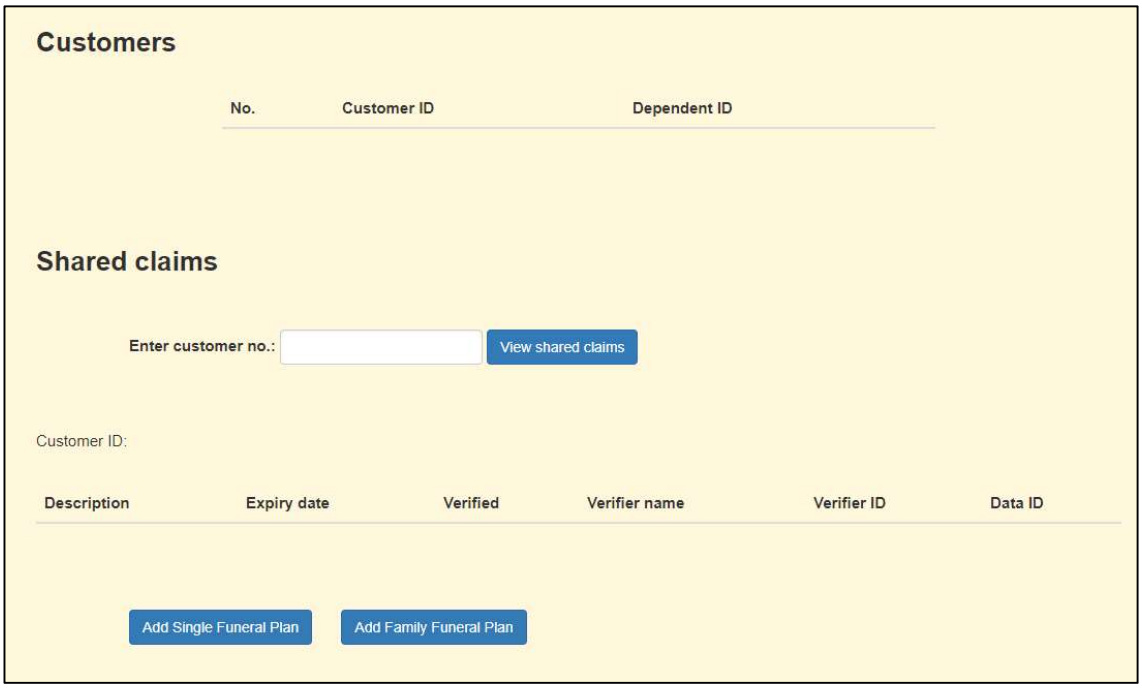


Figure 7. SSI application – Insurer section.

We define a JavaScript program `App` that consists of several methods and global variables for the web3 provider, Ethereum contracts and the user's wallet account address. We also specify a timer to check every 100 milliseconds if a different account is selected in the MetaMask wallet and to update the front-end content accordingly. When the webpage is opened, the program starts and calls the `initWeb3` method to configure a web3 object that connects the application to the local blockchain. We can either use the web3 object and provider already supplied by the MetaMask wallet or create new variables using the Ganache network. Then the `initContract` method is called to add objects for the factory and entity contracts to the `App` object and to set their web3 providers so that users can interact with them. The `updateAccount` method is called and uses the `getCoinbase` function of the Ethereum object in web3 to set the account address stored in the program to the account address currently in the MetaMask wallet. Other functions are also called to listen to events emitted by the contracts and to render content in the front-end. At this stage the program is initialized and the user can now interact with it.

When the user selects an identity to be created in the front-end, the `register` method is called which takes in an integer value for the identity type and invokes the function in the factory contract to create the identity. The types 1, 2, 3 and 4 represent the government, insurer, customer and relative, respectively. An event is emitted and stored in the Ethereum transaction log. The `listenForEvents` method watches for these events and calls the `login` method to update the front-end content.

The `login` method works as follows. Using an instance of the deployed factory contract, we obtain the number of identities. If identities are created for all four entities, then further information is displayed. A method is called to display the services of the government and insurer. From the factory contract, we read the mapping of the user's account address to their identity type. Depending on the type of identity, only the relevant sections in the front-end are shown and methods are called to display claims, requests or customers. The user's identity type and account address are also shown.

The `customerDisplayClaims` method is used to display claims for the customer or his relative. Using an instance of the deployed factory contract, we retrieve the user's contract address and declare an instance of the `IdentityCustomer` contract at that address. We obtain the number of claims with the `getClaimCount` function in the customer contract. The table to output claims in the front-end is cleared. We then loop through each claim by accessing it with the `getClaim` contract function. We combine its elements to form a row and append to the claims table. Note that any field stored as bytes32 in the contract is converted to a string in the application with the `web3.toUtf8` command. The below methods are coded similarly:

- `factoryDisplayServices` displays the array of provider service claims from the factory contract.
- `govDisplayRequests` accesses `getVerifRequestCount` and `getVerifRequest` in the government contract to display its verification requests.
- `corpDisplayCustomers` displays the array of customers stored in the insurer's contract using its `getCustomerCount` and `getCustomer` functions.
- `corpDisplaySharedClaims` is invoked by the insurer from the front-end. The insurer enters a customer number in a form and this value is used in this method to retrieve the array of shared claims for that particular customer from the insurer's contract.

When the customer creates a claim for his proof of ID, he enters data in a form for his identity number, name and birthdate. The method `customerCreateClaimID` captures this information in an array. It accesses the customer's contract and executes the `createClaimID` function with the required inputs. When the `NewClaim` event is emitted, it, obtains and adds the data identifier to the data array. This array is then converted into a JSON data object and saved to the user's device as a plain text file. A JSON object is a collection of data with labels and values. The below methods follow a similar process:

- `customerCreateClaimAddress` is called by the customer to create an address claim.
- `customerCreateClaimAge` is called by the customer to create an age claim.
- `corpCreateFuneralPlan` is invoked by the insurer to create a claim for the customer that he has a particular funeral plan.

When the government verifies a claim from their table of requests, the `govVerifyClaim` method is executed, which calls the `verifyClaim` function in government's contract.

For the customer's interactions with the insurer, the following methods are provided:

- `customerSignUp` calls the `addCustomer` function in insurer's contract to add the customer's account address to their customer array.
- `customerAddDependent` adds a dependent for the customer using the `addDependent` function in the insurer's contract and the dependent's account address entered by the customer in the front-end.
- `customerShareClaim` takes in a claim number from a form and calls the `shareClaim` function in the insurer's contract to share that claim with the insurer.

4. Results and analysis

4.1 Demonstration of the use case

Using illustrations of the application, we demonstrate the identity verification process when the customer applies for a Single Funeral Plan and Family Funeral Plan at different times from the insurer. These services are displayed in the below figure.

Figure 8. Service claims from the government and insurer are displayed in the SSI application.

Suppose it is 15-Februay-2019 and the customer has contacted the insurer's Youth Markets Division to apply for a Single Funeral Plan. Figure 9 shows how he can use the application to sign up as a customer with the insurer and share his verified claims on SA identity, recent residential address and age.

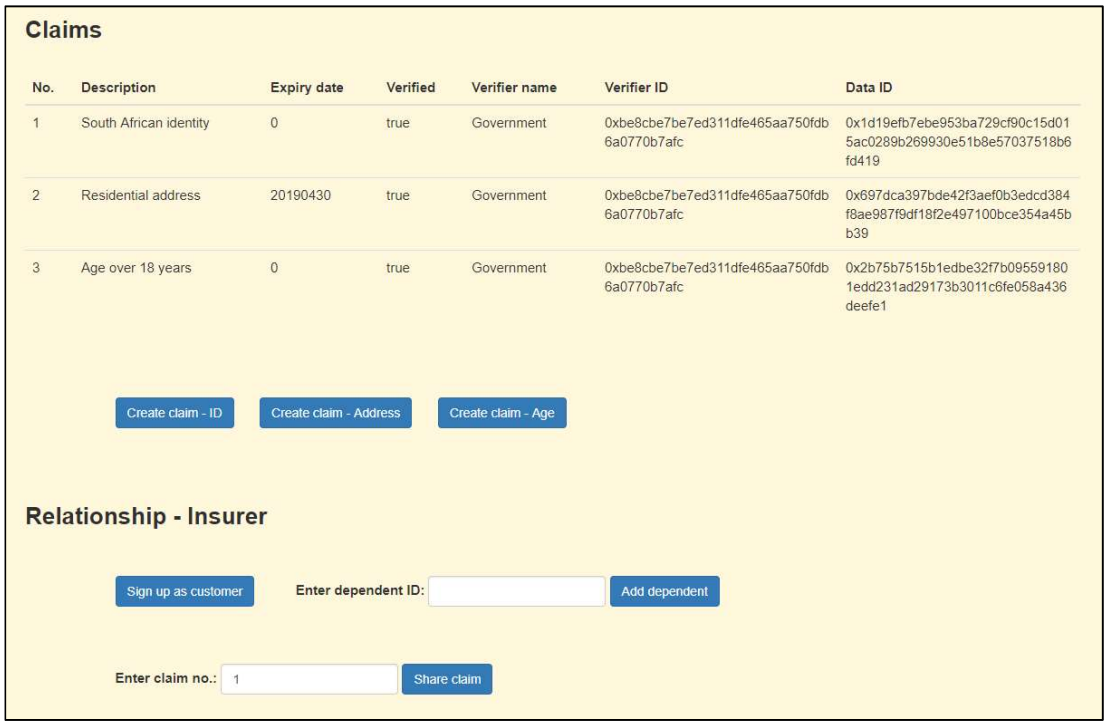


Figure 9. The customer uses the SSI application to sign up to the insurer and share his verified claims on SA identity, recent residential address and age.

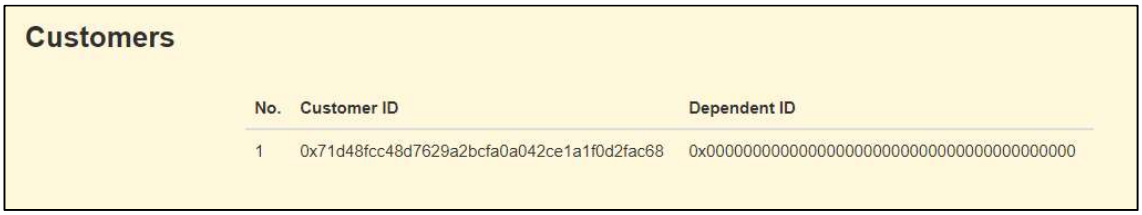


Figure 10. The Youth Markets employee sees that the customer's identifier has been added to the insurer's identity in the SSI application.

The Youth Markets employee accesses the application and sees that the customer's identifier has been added to the insurer's identity (Figure 10). He enters customer number 1 in a form to view the customer's shared claims (Figure 11), and checks that they are verified by government and that the address claim has not expired. A claim on Single Funeral Plan is submitted (Figure 12), added to the customer's shared claims and issued to the customer (Figure 13).

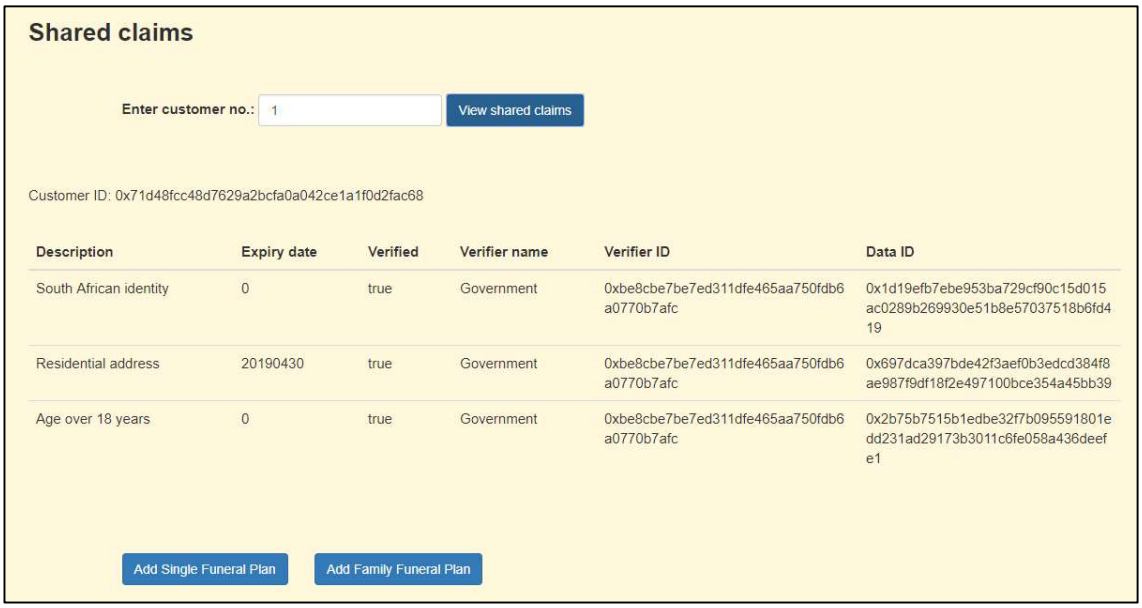


Figure 11. The Youth Markets employee views the customer's shared claims in the SSI application.

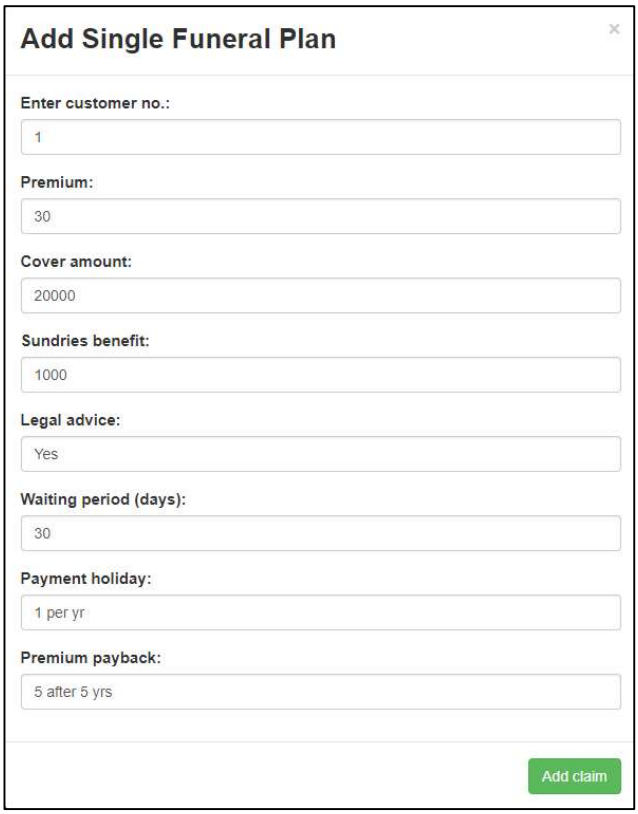


Figure 12. Form in the SSI application used by Youth Markets to issue the customer with a claim on Single Funeral Plan.

Claims

No.	Description	Expiry date	Verified	Verifier name	Verifier ID	Data ID
1	South African identity	0	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0x1d19efb7ebe953ba729cf90c15d015ac0289b269930e51b8e57037518b6fd419
2	Residential address	20190430	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0x697dca397bde42f3aef0b3edcd384f8ae987f9df18f2e497100bce354a45bb39
3	Age over 18 years	0	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0x2b75b7515b1edbe32f7b095591801edd231ad29173b3011c6fe058a436deefe1
4	Single Funeral Plan	0	true	Insurer	0xed2bca8de9581fa9bcc3161abf1d02399a83fd85	0x93d0a6e7bbd26e5cb072ab3f1ce300b5cb1442b1fed648b89994f97d23d29950

Figure 13. The customer views their claim on Single Funeral Plan in the SSI application.

A month later, the customer contacts the insurer's Family Markets Division to apply for a Family Funeral Plan for himself and his relative. He submits his relative's identifier to the insurer using a form shown in Figure 9. The relative accesses the interface similar to Figure 9, signs up to the insurer and shares his claim on SA identity. The Family Markets employee sees that the customer already shared identity claims with the address claim still active and a claim on Single Funeral Plan (Figure 14). He also checks the relative's shared identity claim (Figure 15).

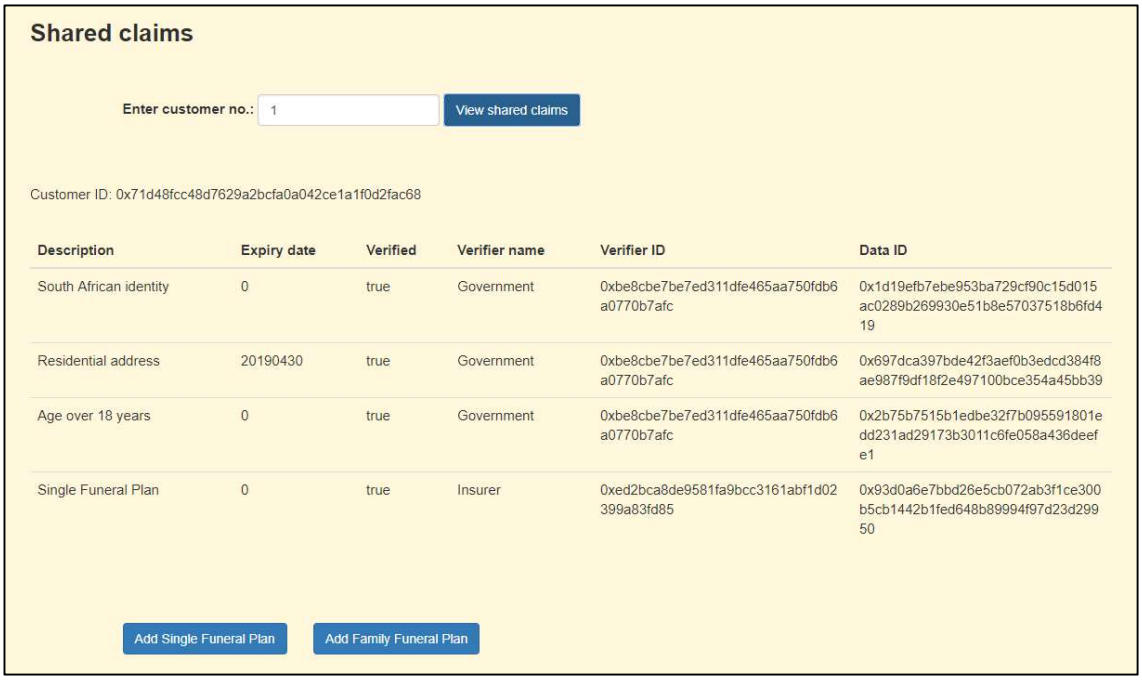
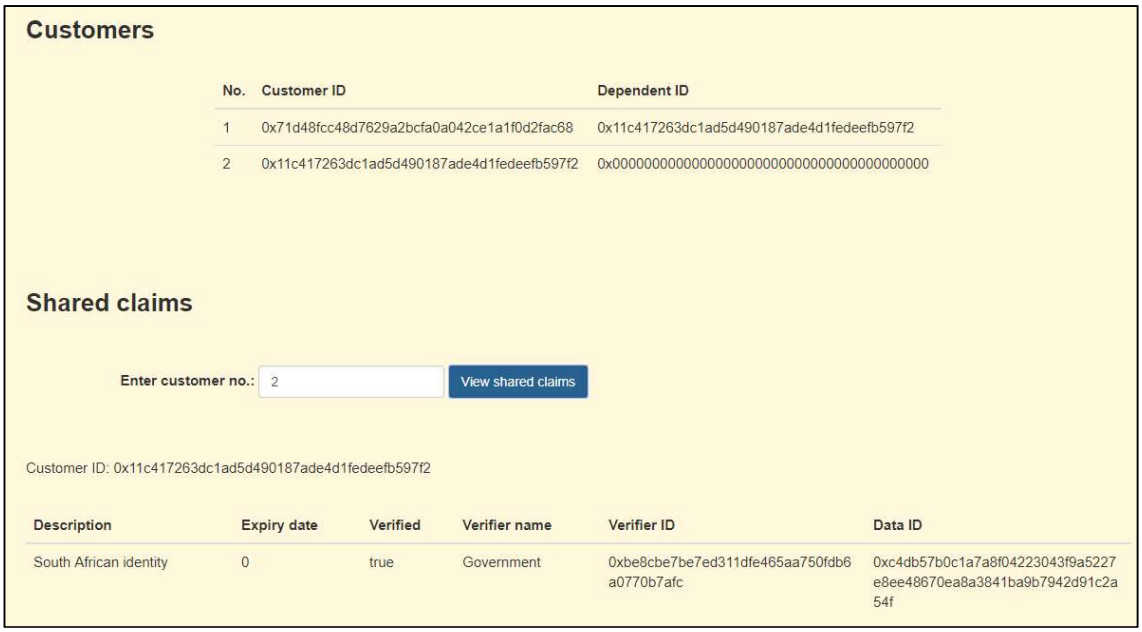


Figure 14. The Family Markets employee views the customer's shared claims in the SSI application.



Customers

No.	Customer ID	Dependent ID
1	0x71d48fcc48d7629a2bcfa0a042ce1a1f0d2fac68	0x11c417263dc1ad5d490187ade4d1fedeefb597f2
2	0x11c417263dc1ad5d490187ade4d1fedeefb597f2	0x0000000000000000000000000000000000000000

Shared claims

Enter customer no.: 2 View shared claims

Customer ID: 0x11c417263dc1ad5d490187ade4d1fedeefb597f2

Description	Expiry date	Verified	Verifier name	Verifier ID	Data ID
South African identity	0	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0xc4db57b0c1a7a8f04223043f9a5227e8ee48670ea8a3841ba9b7942d91c2a54f

Figure 15. The Family Markets employee views the customer's relative's shared claim in the SSI application.

When a suitable Family Funeral Plan is created for the customer, a claim is submitted, added to the customer's shared claims and issued to the customer (Figure 16).

Claims

No.	Description	Expiry date	Verified	Verifier name	Verifier ID	Data ID
1	South African identity	0	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0x1d19efb7ebe953ba729cf90c15d015ac0289b269930e51b8e57037518b6fd419
2	Residential address	20190430	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0x697dca397bde42f3aef0b3edcd384f8ae987f9df18f2e497100bce354a45bb39
3	Age over 18 years	0	true	Government	0xbe8cbe7be7ed311dfe465aa750fdb6a0770b7afc	0x2b75b7515b1edbe32f7b095591801edd231ad29173b3011c6fe058a436deefe1
4	Single Funeral Plan	0	true	Insurer	0xed2bca8de9581fa9bcc3161abf1d02399a83fd85	0x93d0a6e7bbd26e5cb072ab3f1ce300b5cb1442b1fed648b89994f97d23d29950
5	Family Funeral Plan	0	true	Insurer	0xed2bca8de9581fa9bcc3161abf1d02399a83fd85	0x2823560e9550f52a182f2b65d27dee09974492f17242e65deb111ce42ef3ef1b

Figure 16. The customer views their claim on Family Funeral Plan in the SSI application.

4.2 Properties of the system

We comment on the strengths and limitations of the above identity system for funeral insurance with consideration of Allen (2016)'s principles of SSI. Firstly, the system is decentralized as it runs on the public Ethereum blockchain and the customer, for example, accesses it from the application located on his device. Thus, there is no central party that manages the system. Each entity can create their smart contract identity with a unique public address and has independent control of it through their wallet. The address does not contain any personal information, unlike a SA identity number which includes a birthdate. Identities have no time-based restrictions and currently our system does not have the option for an entity to remove their identity.

Provided that there are enough nodes in the Ethereum network, the system will always be operating and accessible to users. Since a user can connect to their identity from any location, it is portable. The customer accesses the system with a wallet such as MetaMask (ConsenSys, 2018b) and needs to be responsible in managing his credentials, as there is no mechanism to recover lost keys. We design the system for a specific use case, however, it can be extended to include other service providers globally and a user can share the same claims with them in a convenient manner. There is also transparency in the system since the source code is open and can be read by others. Despite the benefits of the public nature of the blockchain network, we note the risk that it can be controlled by a node with enough mining resources (Tapscott and Tapscott, 2016).

The data supporting a claim is not stored on the ledger, so a customer has more control of his personal information in comparison to the current centralized identity model. Data used to create a claim is exported as a JSON formatted text file and saved to the user's device. Below is an example of the data object produced when the customer generates a claim on his SA identity.


```
{
   "Data ID":"0x1d19efb7ebe953ba729cf90c15d015ac0289b269930e51b8e57037518b6fd419",
   "ID number":9502010000000,
   "Name":"John Doe",
   "Date of birth":19950201
}
```


Figure 17. JSON data object for the customer's claim on his SA identity.

A drawback of our system is that it does not include a tool for the customer to manage his data. A technical user may store their data in a local database. Non-technical users could benefit from a wallet such as the SelfKey wallet (The SelfKey Foundation, 2017) that provides functionality to manage data locally. Claims about the customer or his relative are

shared with consent to the insurer, and a claim is designed with simple descriptions such as "Age over 18 years" so that the customer shares the least amount of information needed to obtain a funeral plan. A major benefit of our system is the ability of Family Markets to re-use the shared claims provided by the customer. This increases efficiency for both the customer and insurer. The insurer also does not need to store sensitive customer data in their own databases, which reduces their expense for storage space and security (Tobin and Reed, 2017). However, they do store their customers' public identifiers and shared claims in their smart contract.

Privacy is in our system is limited to not having the customer's data shared over the network and stored on the ledger or by other parties. Since the Ethereum blockchain is public, identity transactions, such as interactions between contracts to share claims, can be read by anyone. Although claims are created separately, they are linked to a certain address. For example, a viewer may observe that a particular address has claims on SA identity and residential address even if the underlying data is not recorded. The use of private ledgers would be a way to improve on this issue (Windley, 2016). This is an important limitation of our system and a solution is outside of this project's scope.

4.3 General impact of self-sovereign identity

Given the current processes for identity verification and usage, we consider what impacts the SSI method is expected to have on those processes and its users.

4.3.1 Impact on individuals

Individuals will need to take responsibility for managing their own credentials to their wallet and identity. It will require users to be comfortable in using mobile applications securely. uPort (Lundkvist et al., 2016) offers key recovery through a smart contract, but the process relies on appointing other uPort delegates such as family or friends, and this may not suit every user. Sovrin's system (Windley, 2016) has a different key-pair for every relationship that a user has with a service provider. A user will interact with the network through an agent's application. So the user will manage a private key for access to the application, but will not need to directly manage every private key from their relationships, as the agent's system will manage this.

Dunphy and Petitcolas (2018) comment on the user education and experience of the SSI solutions it reviews. Currently there seems to be adequate information that assists an individual in understanding how to use the technology, but the specifications of how these systems function is more suited to technical users. This may not be an issue though, since users have adopted systems such as Facebook without needing to understand the underlying structure. However, the increased occurrence of breaches into the data stored by large companies have made the public more aware of the security issues faced by traditional systems and the effect it has on their private information (Lingham and Smith, 2018). Thus,

for SSI solutions to be widely adopted, it will require that consumers be made aware of both the benefits and risks associated with these systems.

The advantages of these systems such as quicker verification of information and more user control is certainly appealing to younger consumers who are less inclined than the average consumer to spend time having to acquire a product or service. This behaviour motivates the adoption of SSI technology by companies who will want to remain attractive to the younger market segment. Another benefit to consumers is that businesses will store less of their data which increases their privacy (Tobin and Reed, 2017). However, a possible downside to this is that businesses may be less able to provide a personalised experience to a customer since they will have less data for building models to classify and predict customer behaviour (Xu et al., 2011; James et al., 2013).

4.3.2 Impact on organizations

In terms of how SSI systems are implemented, corporates will need to adapt their views of customer information and need to be comfortable with acquiring identity verification without the actual information itself. For example, an application for funeral insurance that requires a recent proof of address for the customer will only require proof that the customer has a valid address and that it is verified by a reliable party such as a government authority. The actual details of the address will not need to be captured. A corporate will also need to learn to rely on verifications carried out by other departments internally. They may even need to depend on external trusted parties to verify claims, which is an aspect of the SSI system offered by Civic (Lingham and Smith, 2018).

Since identity information can easily be requested through a SSI platform, businesses will not need to store any information. The proof that KYC processes are performed will be recorded on the blockchain supporting the SSI system and this transaction will be available for auditors to check. The increased efficiency in verification of customers will lower the high costs of capturing, storing and securing customer data (Lingham and Smith, 2018).

Currently customer data is used to develop statistical models that assist the business in understanding and managing risks, and in gaining insights into customer behaviour which can help them market their services better or build products more relevant to the consumer (Xu et al., 2011; James et al., 2013). An example of this type of data is credit information captured and stored by credit bureaus (Miller, 2003). This data is supplied to them by service providers who are allowed to do so under current regulations (National Credit Regulator, 2007). An individual provides consent to this process so that they may gain access to credit by being scored (Miller, 2003), but they do not explicitly share each piece of data. In a SSI system where a user has more control over their data, businesses may become restricted in what user information and how much of it they can obtain to develop models. This provides an opportunity where users may be financially rewarded for sharing data and this data could be shared in a way that does not link it to a specific consumer. Businesses may also need to

adapt their models to new types of data being supplied. For example, a user may share their age group rather than their actual age. Furthermore, there are cases where historic customer behaviour data is not essential, such as with funeral insurance. This may create an opportunity to build predictive models using premium payment behaviour shared by users.

5. Conclusion

5.1 Summary

In this project we developed a system to demonstrate the concept of self-sovereign identity for a customer interacting with two funeral insurance departments, Youth Markets and Family Markets, within a company. Self-sovereign identity is a method where statements about an identity can be shared with others without sharing any sensitive data (Tobin and Reed, 2017). When an individual applies for a financial product, documentation must be supplied that proves characteristics such as his national identity, residential address and age. Financial service providers are legally required to capture this information and check its validity (Financial Intelligence Centre Act, 2001), which helps prevent the business from being exposed to financial crimes (Cox, 2014). However, as more information is stored digitally, it increases the cost to the business for reserving space for this data and securing it (Lingham and Smith, 2018). With a self-sovereign identity model, the business does not need to collect personal customer data, but only a claim that the customer has a certain attribute and that it is verified by an appropriate authority (Windley, 2016).

Our system achieves this method using Ethereum-based smart contracts (Wood, 2014; Buterin, 2014) to create, store and transfer identity claims. The identity transactions are appended to a blockchain, which cannot be historically edited, and so there will always be a record of such events for auditing purposes. We use a local development blockchain Ganache (ConsenSys, 2018a) and a MetaMask wallet (ConsenSys, 2018b) for submitting transactions from accounts. We also built a browser-based application from which users can transact with the smart contracts. The customer has a contract that enables him to create claims for his identity, address and age, and to get them verified by the government, which also has its own contract. The customer can then share these claims with Youth Markets to obtain a Single Funeral Plan. The insurer stores these claims in its smart contract so that they can be viewed when the customer applies for another policy, a Family Funeral Plan, from Family Markets which accesses the same smart contract. Furthermore, the customer receives verified claims of his funeral plans from the insurer.

This system is beneficial to the business since there is less duplicated work and less sensitive data needs to be stored and secured, which should reduce their Know-Your-Customer costs (Lingham and Smith, 2018). The customer also has a better user experience and more privacy and control of the data used to generate his claims (Dunphy and Petitcolas, 2018), since the data is saved to his device and not stored on the ledger. However, privacy is not complete in our system since the identity transactions are viewable on the public blockchain and can be linked to specific users. Another disadvantage is the lack of a data management tool for the customer. Despite these issues, the sharing of claims without revealing personal information is a significant improvement over the centralized identity model.

5.2 Further work

The above self-sovereign identity system focussed on the customer application for funeral cover. This system could be adapted to show how the processes to register a death with government and submit a claim for an insurance pay-out can be improved. C2Legacy (Tichler, 2018) aims to provide a system on Ethereum where one's digital assets can be transferred more seamlessly to beneficiaries in the event of a validated death. Part of their system involves verifying a death using information from a doctor, coroner, government or legal professional, and issuing a "blockchain-based death certificate". However, that method of recording death is not verified by a trusted authority and would not be usable by insurance providers who need legal death reports and death certificates.

Cognizant (2017) has explored how a private permissioned blockchain and smart contracts can be used to make the processes mentioned above more convenient and efficient for beneficiaries. In most cases around the world, the current system involves a physician or coroner certifying a death notification and a government authority providing the death certificate which is then used to manually claim insurance. Cognizant (2017) proposes the following design that involves a few entities interacting with a multiparty smart contract. When a death occurs, it is recorded by a hospital on the blockchain. The insurer's smart contract recognizes the event and notifies the specified beneficiary to select a funeral home with which to register the death. The funeral home is notified and receives a form that already contains information on the deceased. The form is completed and submitted to government which issues a death certificate and burial permit to the beneficiary. These documents are shared with the funeral home in order to proceed with the funeral and shared with the insurer in order to disburse the insurance cover. The data is stored on IPFS or a similar system and linked hashed identifiers are stored on the blockchain. This approach could be made more self-sovereign where the data is shared off-chain and only verified claims containing the encrypted identifiers to the data is stored on the ledger. Since funeral cover benefits do not generally depend on information such as the cause of death, verified claims on a death can then be shared with the funeral insurance provider without revealing the underlying data.

References

Ahn, G.J., Ko, M. and Shehab, M. (2009). Privacy-Enhanced User-Centric Identity Management. *IEEE International Conference on Communications*, Dresden, 2009, 1-5.

Allen, C. (2016). *The Path to Self-Sovereign Identity* [online]. Available at: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html (Accessed: January 2019)

The Banking Association South Africa. (2018). *Treating Customers Fairly* [online]. Available at: https://www.banking.org.za/consumer-information/legislation/treating-customers-fairly (Accessed: December 2018)

Bertram, S. (2018). *Fintech and Cryptocurrencies Wiki* [online]. Available at: https://github.com/cogeorg/teaching/wiki/Fintech-and-Cryptocurrencies (Accessed: July 2018)

Bertram, S. and Georg, C.P. (2018). *A privacy-preserving system for data ownership using blockchain and distributed databases* [online]. Available at: https://arxiv.org/abs/1810.11655 (Accessed: September 2018)

Birch, D.G.W. (ed.) (2007). *Digital Identity Management*. Surrey: Gower Publishing Limited.

Buterin, V. (2014). *A next-generation smart contract and decentralized application platform* [online]. Available at: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf (Accessed: January 2019)

Case, A., Garrib, A., Menendez, A. and Olgiati, A. (2013). Paying the Piper: The High Cost of Funerals in South Africa. *Economic Development and Cultural Change*, 62(1), 1-20.

Cognizant. (2017). *Blockchain: A Potential Game-Changer for Life Insurance* [online] Available at: https://www.cognizant.com/content/dam/Cognizant_Dotcom/whitepapers/blockchain-a-potential-game-changer-for-life-insurance-codex2484.pdf (Accessed: December 2018)

ConsenSys. (2018a). *Ganache* [online]. Available at: https://trufflframework.com/ganache (Accessed: November 2018)

ConsenSys. (2018b). *MetaMask* [online]. Available at: https://metamask.io/ (Accessed: November 2018)

ConsenSys. (2018c). *Truffle* [online]. Available at: https://trufflframework.com/truffle (Accessed: November 2018)

Corbet, S., Lucey, B. and Yarovya, L. (2018). Datestamping the Bitcoin and Ethereum Bubbles. *Finance Research Letters*, 26, 81-88.

Cox, D. (2014). *Handbook of Anti-Money Laundering*. UK: John Wiley & Sons.

Cummins, J.D. (1991). Statistical and Financial Models of Insurance Pricing and the Insurance Firm. *The Journal of Risk and Insurance*, 58(2), 261-302.

Department of Home Affairs. (2018a). *Death Certificates* [online]. Available at: http://www.dha.gov.za/index.php/civic-services/death-certificates (Accessed: December 2018)

Department of Home Affairs. (2018b). *Identity Documents* [online]. Available at: http://www.dha.gov.za/index.php/civic-services/identity-documents (Accessed: September 2018)

Diffie, W. and Hellman, M.E. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6), 644-654.

Diffie, W. and Hellman, M.E. (1979). Privacy and Authentication: An Introduction to Cryptography. *Proceedings of the IEEE*, 67(3), 397-427.

Dunphy, P. and Petitcolas, F.A.P. (2018). A First Look at Identity Management Schemes on the Blockchain. *IEEE Security and Privacy,* 16(4), 20-29.

Edwards, B., Hofmeyr, S. and Forrest, S. (2016). Hype and heavy tails: A closer look at data breaches. *Journal of Cybersecurity*, 2(1), 3-14.

Encyclopaedia Brittanica. (2016). *Numeral System* [online]. Available at: https://www.britannica.com/science/numeral-system (Accessed: December 2018)

Ethereum. (2016). *web3.js - Ethereum JavaScript API* [online]. Available at: https://web3js.readthedocs.io/en/1.0/index.html (Accessed: December 2018)

Ethereum. (2018). *Solidity Documentation Release 0.4.24* [online]. Available at: https://readthedocs.org/projects/solidity/downloads/pdf/v0.4.24/ (Accessed: December 2018)

Financial Intelligence Centre. (2005). *Guidance Note 3A: For accountable institutions on client identification and verification and related matters* [online]. Available at: https://www.fic.gov.za/Documents/Forms/DispForm.aspx?ID=26 (Accessed: December 2018)

Financial Intelligence Centre Act. (2001). [online]. Available at: https://www.fic.gov.za/Documents/FIC%20Act%20with%202017%20amendments%20(1)%20(1).pdf (Accessed: December 2018)

Garfinkel, S. (1995). *PGP: Pretty Good Privacy*. USA: O'Reilly & Associates, Inc.

Gatteschi, V., Lamberti, F., Demartini, C., Pranteda, C. and Santamaría, V. (2018). Blockchain and Smart Contracts for Insurance: Is the Technology Mature Enough?. *Future Internet*, 10(2), 20.

Grüner, A., Mühle, A. and Meinel, C. (2018). *On the Relevance of Blockchain in Identity Management* [online]. Available at: https://arxiv.org/pdf/1807.08136 (Accessed: September 2018)

Higgs, E. (2011). *Identifying the English: A History of Personal Identification 1500 to the Present.* New York: Continuum Books.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. New York: Springer.

Liberty. (2018). *Funeral benefits and plans* [online]. Available at: https://www.liberty.co.za/our-offering/family/Pages/funeral-benefits-and-plans.aspx (Accessed: December 2018)

Lingham, V. and Smith, J. (2018). *Civic white paper* [online]. Available at: https://tokensale.civic.com/CivicTokenSaleWhitePaper.pdf (Accessed: October 2018)

Lundkvist, C., Heck, R., Torstensson, J., Mitton, Z. and Sena, M. (2016). *uPort: A Platform for Self-Sovereign Identity* [online]. Available at: http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf (Accessed: September 2018)

Maler, E. and Reed, D. (2008). The Venn of Identity: Options and Issues in Federated Identity Management. *IEEE Security & Privacy*, (2), 16-23.

Mesropyan, E. (2017). *22 Companies Leveraging Blockchain for Identity Management and Authentication* [online]. Available at: https://gomedici.com/22-companies-leveraging-blockchain-for-identity-management-and-authentication/ (Accessed: September 2018)

Miller, M. J. (ed.). (2003). *Credit Reporting Systems and the International Economy*. USA: The MIT Press.

Modi, R. (2018). *Solidity Programming Essentials*. Birmingham: Packt Publishing.

National Credit Regulator. (2007). *The National Credit Act, 2005, Volume 1: 2007* [online]. Available at: https://www.ncr.org.za/documents/pages/ENGLISH.pdf (Accessed: January 2019)

Node.js Foundation (2018). *Node.js* [online]. Available at: https://nodejs.org/en/ (Accessed: November 2018)

Offerman, A. (2018). *Swiss City of Zug issues Ethereum blockchain-based eIDs* [online]. Available at: https://joinup.ec.europa.eu/document/swiss-city-zug-issues-ethereum-blockchain-based-eids (Accessed: November 2018)

Old Mutual. (2018). *Funeral Cover* [online]. Available at: https://www.oldmutual.co.za/personal/funeral-cover (Accessed: December 2018)

Papp, R. and Matulich, E. (2011). Negotiating the deal: using technology to reach the millennials. *Journal of Behavioral Studies in Business*, 4, 1 [online]. Available at: https://www.aabri.com/manuscripts/111063.pdf (Accessed: February 2019)

Protection of Personal Information Act. (2013). [online]. Available at: https://www.gov.za/sites/default/files/gcis_document/201409/3706726-11act4of2013protectionofpersonalinforcorrect.pdf (Accessed: December 2018)

Richters, O. and Peixoto, T.P. (2011). Trust transitivity in social networks. *Public Library of Science One*, 6(4), e18384.

Roth, J. (2000). *Informal micro-finance schemes: The case of funeral insurance in South Africa*. Geneva: International Labour Office.

Rothstein, A. (2017). *New Scientist: The End of Money*. London: John Murray Learning.

Russell, M.A. (2014). *Mining the Social Web*. USA: O'Reilly Media, Inc.

Sanlam. (2018). *Sanlam My Choice Funeral Plan*s [online]. Available at: https://www.sanlam.co.za/personal/insurance/personal-and-family-insurance/funeral-cover/Pages/mychoice.aspx (Accessed: December 2018)

Sironi, P. (2016). *FinTech Innovation: From Robo-Advisors to Goal Based Investing and Gamification*. UK: John Wiley & Sons.

Skan, J., Dickerson, J. and Masood, S. (2015). The Future of Fintech and Banking: Digitally disrupted or reimagined?, *Accenture* [online]. Available at: https://www.accenture.com/_acnmedia/Accenture/Conversion-Assets/DotCom/Documents/Global/PDF/Dualpub_11/Accenture-Future-Fintech-Banking.pdf (Accessed: February 2019)

The SelfKey Foundation. (2017). *SelfKey* [online]. Available at: https://selfkey.org/wp-content/uploads/2017/11/selfkey-whitepaper-en.pdf (Accessed: February 2019)

Sovrin Foundation. (2018a). *Sovrin: A Protocol and Token for Self-Sovereign Identity and Decentralized Trust* [online]. Available at: https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf (Accessed: July 2018)

Sovrin Foundation. (2018b). *Stewards* [online]. Available at: https://sovrin.org/stewards/ (Accessed: December 2018)

Swan, M. (2015). *Blockchain: Blueprint for a new economy*. USA: O'Reilly Media, Inc.

Tapscott, D. and Tapscott, A. (2016). *Blockchain Revolution*. UK: Portfolio Penguin.

Tarkowski Tempelhof, S., Teissonniere, E., Fennell Tempelhof, J. and Edwards, D. (2017). *Bitnation, Pangea Jurisdiction and Pangea Arbitration Token (PAT): The Internet of Sovereignty* [online]. Available at: https://github.com/Bit-Nation/Pangea-Docs/raw/master/BITNATION%20Pangea%20Whitepaper%202018.pdf (Accessed: November 2018)

Tichler, M. (2018). *C2Legacy* [online] Available at: https://c2legacy.io/white-paper/ (Accessed: December 2018)

Tobin, A. (2017). *Sovrin: What Goes on the Ledger* [online]. Available at: https://www.evernym.com/wp-content/uploads/2017/07/What-Goes-On-The-Ledger.pdf (Accessed: December 2018)

Tobin, A. and Reed, D. (2017). *The Inevitable Rise of Self-Sovereign Identity* [online]. Available at: https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf (Accessed: October 2018)

Tykn B.V. (2018). *Ana: An Overview* [online]. Available at: https://docs.google.com/document/d/1epIP5nWZr3GF-1dWVjE6JeMJFfFU8_OOcF5ufMcVZNY/edit (Accessed: December 2018)

uPort. (2018). *IdentityManager* [online]. Available at: https://github.com/uport-project/uport-identity/blob/develop/docs/reference/identityManager.md (Accessed: January 2019)

Windley, P.J. (2016). *How Sovrin Works* [online]. Available at: https://sovrin.org/wp-content/uploads/2018/03/How-Sovrin-Works.pdf (Accessed: October 2018)

Wood, G. (2014). *Ethereum: A secure decentralised generalised transaction ledger* [online]. Available at: http://gavwood.com/paper.pdf (Accessed: December 2018)

World Economic Forum. (2018). *Identity in a Digital World: A new chapter in the social contract* [online]. Available at: http://www3.weforum.org/docs/WEF_INSIGHT_REPORT_Digital%20Identity.pdf (Accessed: November 2018)

Xu, H., Luo, X. R., Carroll, J. M. and Rosson, M. B. (2011). The personalization privacy paradox: An exploratory study of decision making process for location-aware marketing. *Decision support systems*, 51(1), 42-52.