

Identity Management Service Based on Blockchain and Smart Contracts to Recover Dynamic Identities and Multiple Identities

N.M.S.I.Jinathissa
Index No: 16440483

April 2019

Abstract

With modern technological advancements, people tend to represent themselves digitally, and that lead to the concept of "Digital Identity." A digital identity represents a real-world entity and its properties (so-called identifiers) so that it can be uniquely identified digitally without the physical presence. These entities can either be individuals or organizations as well.

IdM services emerged in order to manage these digital identities. They try to map the complicated real-world identities into digital identities, keeping track of the authenticity of data circulated. This is not as simple as it looks, and not every IdM service is as secure as it seems.

One of the most reliable and secure implementations for such services is considered the decentralized blockchain based IdM services, due to their peer-to-peer authentication and global availability. In this research, we are proposing a robust lost identity recovering approach to implement in a decentralized IdM service which is having the capability of handling dynamic and multiple identities.

Contents

1	Introduction	1
1.1	Problem Domain	2
1.1.1	Multiple and Dynamic Identities	2
1.1.2	Digital Identity	3
1.1.3	Identity Management (IdM) in Cyberspace	3
1.2	Motivation	5
1.3	Aims and Objectives	6
1.4	Methodology	7
1.4.1	Scope and Delimitations	7
1.5	Outline	8
2	Literature Review	9
2.1	Key concepts of Blockchain based IdM Services	9
2.1.1	Blockchain	9
2.1.2	Transactions	10
2.1.3	Proof-of-Work	10
2.1.4	Accounts	10
2.2	IdM Solutions Available	10
2.3	Problem Analysis	12

2.4	Summary	13
3	Design	14
3.1	Methodology	16
4	Implementation	19
4.1	Development Environment	20
4.2	Selection of the Tool and Libraries	20
4.2.1	Truffle Framework	20
4.2.2	Ganache	20
4.2.3	Node Package Manager (npm) bundled with node.js	21
4.2.4	Chrome Browser along with Metamask extension for Google Chrome	21
4.3	Implementation of the Basic IdM Service	22
4.4	Implementation of the Lost Identity Recovering Feature	25
4.5	Optimization of the IdM Framework	27
5	Evaluation	28
5.1	General Knowledge on the concept of "Identity"	29
5.2	Usage of Available IdM Services	30
5.3	Comparison of the Developed DApp with Available IdM Service	31
6	Conclusion	33
6.1	Limitations of the Provided Solution	33
6.2	Future Work	34
6.3	Summery	34

List of Figures

3.1	<i>Smart contracts for the proposed IDM service [5]</i>	15
3.2	<i>Application Design</i>	17
4.1	<i>Web interface for designation verification</i>	23
4.2	<i>Web interface for designation verification</i>	24
4.3	<i>User Profile</i>	24
4.4	<i>IdM Service Registration</i>	25
4.5	<i>Request for Identifier Recovery</i>	26

List of Tables

5.1	Compression Analysis	31
5.2	Competative Scores	32

List of Abbreviations

IdM	Identity Management
DApp	Distributed Application
IoT	Internet of Things
IoA	Internet of Agreements
EOA	Externally Owned Accounts
GUI	Graphical User Interface
CLI	Command Line Interface
npm	Node Package Manager
I/O API	Input/Output Applications Programming Interface

Chapter 1

Introduction

Identities in cyberspace (also known as the "Digital Identities") represent the real-world entities (such as individuals and organizations) related to the application domains [2]. An identity consists of identifiers; a set of characteristics of the entity which are used for its identification purpose [3].

The identifiers may or may not be unique within the identity domain (ex: Social Security number vs. date of birth). They may or may not stay constant during the lifespan of a particular identity (ex: date of birth vs. designation). The possible characteristics of a particular identity may also differ depending on the type of real-world entity being identified. For example, a "Date of Birth" applies to people; however, it does not apply to organizations. What applies to the organizational contexts are "Founded Date" or a "Registered Date," depending on the domain. On the other hand, a "Company Registration Number" applies to a company, but not to a person. A person will have a "Social Security number (SSN)" or "National Identification Card (NIC) number."

As "Digital Identity" and the "Internet of Agreements" (IoA) [1] became concerning aspects in the cyberspace, Identity Management(IdM) frameworks started emerging. IdM systems are being developed and implemented adapting several features to map the virtual identities to the real-world identities [15] [17]. Since the real-world identities are too complicated to convert into digital identities as they are [2], it is challenging to implement an IdM system fully outfitted to administer all the features of a real identity.

For example, some identifiers in real-world identities are potential to get changed over time (ex: designation) while some identifiers (ex: date of birth) stay constant. A person shall have multiple identities related to him/ her with respect to a single domain. For example, a bank employee of a particular bank can also hold

personal accounts in the same bank, making him/ her a customer of the bank.

Nevertheless, the real challenge arises as these IdM services are designed to be used by the general public in order to maintain their digital identities. In this case, "recovering a lost identity" if the user loses his/ her credentials to access the service is the new problem. This is more than just resetting usernames and passwords since the system should eliminate potential identity theft. Other than that, none of the identities can be repeated in the trusted network (which is the IdM framework) as there should not be any duplicate identities.

With this project, our goal is to address the issues in recovering the lost identities in an IdM framework which can handle multiple identities and dynamic identities.

1.1 Problem Domain

1.1.1 Multiple and Dynamic Identities

One's identity in cyberspace is an essential and critical aspect in the modern world since people have to digitally represent themselves in a variety of different way similar to the distinctiveness of the human identity in the physical world [1]. Nowadays, each person has the opportunity of living multiple lives in parallel, both in the real world as well as in the virtual world, therefore allowing to have multiple identities in parallel. As mentioned in the above example, let us think of a person who works for a bank (thus, an employee of the bank), having his/ her personal savings account in the same bank (thus, a customer of the bank). In this case, the person holds two identities with respect to the bank, as an employee and a customer. Similarly, in the banking system (application domain of the bank), he has **multiple** identities.

An individual's identity transforms continuously throughout the lifetime. An identifier of an individual's identity can get changed in various scenarios. For example, let us consider the 'designation' of an individual. Initially, there will not be a designation assigned to a person at the time of birth. Once he or she gets employed, he or she should be able to get identified from his or her designation. In a complex scenario, a person whose designation is **A** in the company **X** can be promoted or demoted to the designation **B** within the same company. If he or she leaves the company **X** and joins the company **Y**, to the position of **C**, again the designation gets changed. In other words, what we can say that the identities are **dynamic**, that they do not remain the same once identified.

1.1.2 Digital Identity

Computer systems use digital identities to represent external entities. An external entity can be a person, an organization, an application, or a device. ISO/IEC 24760-1 defines identity as “a set of attributes related to an entity. [5]”. These attribute values in a real-world identity are mapped into identifiers in the corresponding digital or virtual identity.

Digital identity tends to become more and more important to the individuals and organizations as the businesses and administrations started to computerize the work done. The advancements in IoT and IAT services and the wide-spread of internet usage around the globe raised the necessity of representing one’s identity digitally since the possibility of involving in frauds is higher when there is no physical presence of any individual in a transaction carried out in the cyberspace. In order to establish a particular transaction with and only with the exact parties that are intended, each party should clearly verify their identity. This verification is achieved in the cyberspace via their digital identities.

1.1.3 Identity Management (IdM) in Cyberspace

Identity Management (IdM) [4] is the set of business processes and supporting infrastructure for the creation, maintenance, and use of digital identities within a legal and policy context. It ensures that the right users are allowed to access the right resources. In other words, authenticates the users to access authorized resources, by verifying each party’s involvement with the information. It covers the scope of how the users gain the identities, how the system protects those identities and the technologies that ensure both of these tasks [5].

Making real-world information accessible to the services on the internet in order to increase the interactions between real and virtual worlds is a trend that started in the research community and now spread in commercial offerings. An area of particular interest within the domain is the management of diverse identities, where “identity” is considered in a broad sense. Issues related to this area include anonymity, pseudonymity, unlinkability, and unobservability [6].

IdM frameworks emerged in order to overcome these issues. They are designed to manage and use the digital identities effectively so that one’s identity is not shared with any unwanted parties and only the required details which have already been verified are shared. Based on the application domain, it might require to allow multiple identities for a single entity simultaneously. As per the previous example, in an application domain of the bank, an individual entity has two identities as a customer

and an employee, both at the same time. Nevertheless, an individual may, of course, have different identities in different domains. For example, a person may have one identity associated with being a customer in a bank and another identity associated with being a student of a particular university.

With the emerge of these IdM services, then came the concern of recovering the lost identities, which is a common issue in any user-involving system. When a user loses the access to the IdM service which stores all the authenticated details, it is impossible to recover them without a proper method that is pre-defined so that no data is exposed to any unauthorized party nor any data on the identity is lost or manipulated during the recovery. These lost identity recovery methods vary from the method of IdM implementation.

In this research, our objective is to develop a sustainable lost identity recovery method to an IdM service which can handle the dynamic and multiple identities. This is relatively a new subject area since some of the systems that already support lost identity recovery; such as uPort and Sovrin do not imply a proper way of recovering the history of a dynamic identity or recovering simultaneous multiple identities at once [15] [16]. The implementation is on Ethereum blockchain based distributed ledger technology and smart contracts, which will be briefly described as follows.

Blockchain-a distributed ledger technology

The word blockchain traces back to Satoshi Nakamoto's original Bitcoin white paper from 2008 [7]. Even though the word "blockchain" has not explicitly mentioned in the paper, it depicts a technology that fundamentals the cryptocurrency as a series of data blocks that are cryptographically chained together. In the broader view, blockchain is a distributed database that provides an unalterable public record of digital transactions. It can be viewed as a distributed digital ledger containing a chain of blocks of information, where a cryptographic signature identifies each block. These blocks are all back-linked; that is, they refer to the signature of the previous block in the chain and that chain can be traced all the way back to the very first block created. Moreover, the Blockchain contains an un-editable record of all the transactions made. The transparent and decentralized nature of the Blockchain network enables the development of a non-refutable and unbreakable record of data, which is the fundamental feature of many applications, such as identity management [8].

Blockchain-based [9] IdM services are decentralized services with no third-party authentication. Instead of centralized authentication, they prefer peer-to-peer identity validation. This decentralization of authentication can be considered significantly important in strengthening IoT security. Such systems have

already been used to securely store information about identity, credentials, and digital rights [10].

Ethereum Blockchain

Ethereum is a **decentralized platform that runs smart contracts** which run on a custom built **blockchain** [18]. Ethereum enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middleman or counter-party risk [18].

Smart Contracts

The term 'Smart Contract' was coined by Nick Szabo in 1994 to identify "a computerized transaction protocol that executes the terms of a contract" [11]. Szabo suggested translating contractual clauses (collateral, bonding, etc.) [12] into the code to minimize the need for trusted intermediaries between transacting parties and the occurrence of malicious or accidental exceptions [13].

These smart contracts can be triggered according to any given conditions. For example, if a new user registers to a smart contract based IdM service and if that service demands the user to nominate the peers within three days, a smart contract can be written to trigger after the third day to prevent that user from nominating any more peers.

1.2 Motivation

In the domain of IdM, losing an identity implies "a user forgets his or her personal ID; thus the access to his or her verified identity is restricted." Without the personal ID, he or she will not be able to prove the identity since he or she is the only one who has access to share the verified credentials to any other party. In such a case, recovering and re-accessing the stored information may result in several issues since the data has to be validated, verified and also needs to be up-to-date.

In blockchain-based IdM services, there are techniques suggested [14] and implemented [15] [16] to recover the identities in a case of losing them. Often this is carried out through the user trusted individuals (peers) who can verify the lost identity. The user nominates these peers at the registration to the IdM service, and smart contracts are used to keep the identity details and trust between the two parties.

Recall the example of dynamic identities mentioned previously in the chapter. The person whose designation is **A** in the company **X** is promoted or demoted to the designation **B** within the same company. In a blockchain-based IdM service, a peer who is nominated by the person should be able to verify his or her designation. At this point, the designations **A** and **B** can be verified by the employer from company **X**. When he or she leaves the company **X** and joins the company **Y**, to the position of **C**, again the designation gets changed. Now the employer from **X** can no longer verify the corresponding designation; hence **X** cannot be a peer anymore. This confirms that the transformations in the identities can affect the trust relationships with peers in the blockchain-based IdM services since the already verified identifiers in the blockchain will no longer be valid due to certain transmutations in the credentials. Hence there should be an effective way to handle these identity changes in blockchain-based IdM services in order to maintain the desired trust in the information getting shared.

Moreover, even the identity is lost and recovered after the person joined the company **Y**, the history of how the designation changed over time should be there since the previous employment of the person is also a part of his or her identity. This means that when recovering the identity, the designations **A** and **B** which were previously verified by the employer from company **X** should also be included in the recovered identity with correspondent data.

Other than that, it is required to recover all simultaneous identities related to the recovering personal ID as there can be consequences if a particular identity is recovered while another identity is still lost and no multiple related identities can be found. As explained previously, suppose that the bank employee who is also a bank customer loses his identity and was able to recover only the identifiers related to his employment and the customer identity is still lost. In such a scenario, the banking application can no longer transfer his salary to the savings account as the account details are not verified that it belongs to the same person that is being employed by the bank. Therefore, an IdM system that supports handling multiple identities simultaneously should be able to smoothly recover all the parallel identities in case of identity lost.

1.3 Aims and Objectives

This research aims to address the issues mentioned above in recovering the lost identities in an IdM service which can handle multiple identities and dynamic identities so that the service is able to recover the history of how an identifier of a user has changed over time, and the simultaneous identities of the individual as well.

In order to achieve the above goal, the following specific objectives should be achieved.

- Implement a basic IdM service with required smart contracts and triggers in Ethereum platform
- Design additional smart contract/s to handle the transforming or dynamic identities
- Experiment on how to incorporate the new smart contract/s to an already existing IdM service
- Recommend a recovery methodology that can recover both dynamic and multiple identities
- Compare the performances and evaluate the model

1.4 Methodology

Project starts with the study of existing IdM services and their ability to handle dynamic and multiple identities and recover any lost identity. After analyzing the available procedures and the compatible technologies, the most suitable approaches to implement the IdM service are selected along with reliable technologies.

1.4.1 Scope and Delimitations

The base concept for the development of IdM service and the smart contract will be the proposal "A Blockchain-based Identity Management Service Supporting Robust Identity Recovery" by Wookun Lee et al. [17]. The proposal provides a solid foundation for a robust identity recovery in a blockchain-based IdM service, but not aimed towards the ability of the IdM system to handle the multiple and dynamic identities. Therefore, the scope of this project will be an extension of the above research so that the IdM framework that handles the multiple and dynamic identities recovers the lost identities.

The application will be a separated DApp [16], and will not be a plugin for any existing DApp or a framework. Therefore, one of the deliverables will be the IdM system and the lost identity recovery will be a feature of the IdM system itself.

1.5 Outline

Chapter 1, the introduction chapter gives an overall idea of the project domain. This included the project motivation, research aims and objectives, the methodology and the scope of the project.

The remaining chapters are organized as the following.

Chapter 2 lays out the literature review for the background study and related work with outlined theories and other concepts. This review intends to identify the gaps among the related works done in several places.

Chapter 3 describes the overall research design and further explanations. Components of the research design are analyzed, and the justifications for the selected methodologies and designs are also explained in chapter 3.

Throughout Chapter 4, it includes the implementation details of the desired design with the discussions of the used technologies and services.

Chapter 5 presents the results for the evaluation process of the aspired design. These results are analyzed with the details of the evaluation criteria for reliable explanations.

Finally, Chapter 6 concludes the overall work done through this research with a summary of the work carried out. It follows the suggestions to the future work for further enhancements of the proposed approach.

Chapter 2

Literature Review

When taking the "Digital Identity" into consideration, Identity Management is most discussed and concerned in the present research community. With the improvements of the IoT and IoA[2] services, decentralized IdM services seem to get more and more interested [10]. There are several IdM services up and running at the moment. They offer a variety of services related to identity management. As the main focus of this project is aimed towards the lost identity recovery in an IdM framework, the most significant services that allow identity recovery will be discussed and their implementation will be analysed in this chapter.

2.1 Key concepts of Blockchain based IdM Services

2.1.1 Blockchain

As mentioned in a previous section, Blockchain [7] is a publically distributed immutable ledger. Blockchain is commonly used for storing application-specific transactions in the present. The concept behind this is, the network selects a collection of random transactions from the pool of unconfirmed transaction, and create groups called "blocks." Here the transactions are arranged according to the timestamp. These blocks are linked to one another in a time-related chain, naming the data structure: **Blockchain**.

The immutability is obtained in blockchain by containing the hash value of the previous block at the next block. In order to add a new block to the blockchain, it is required to solve a cryptography non-reversible hash function. The node that successfully solves it adds the new block and broadcasts the new block to the network, verifying the validity of the block and all the transactions it contains.

2.1.2 Transactions

A transaction [7] has to be signed by the sender using his/ her private key. A new transaction always contains the hash value corresponding to the previous transaction, and the new owner's (receiver's) public key and the signature of the previous owner as mentioned above.

2.1.3 Proof-of-Work

"Proof of work"[7] is the concept that defines the task (which is also known as mining) performed by the nodes or the miners when creating a new group of transactions as a block on to a blockchain. This is done by scanning for a value, when hashed, the hash begins with a series of zero bits. The work required can be verified by executing a single hash [7].

2.1.4 Accounts

To execute transactions on a blockchain based network, accounts are needed. These accounts are referred by their addresses, which are also hash values. Ethereum accounts are of two types [18]; *Externally Owned Accounts (EOAs)* and *Contract Accounts*.

EOAs are the accounts that can transfer Ether (cryptocurrency that is used in the Ethereum framework) to other accounts. Hence they have an ether balance associated with the account. Contract accounts too have an ether balance associated along with an executable code. Either transactions or other contracts can trigger this code. When receiving a transaction, the code

2.2 IdM Solutions Available

uPort [15] and Sovrin [17] are identified as the two most commonly used commercial IdM frameworks based on decentralized applications that support lost identity recovery.

uPort [15] "is a secure, system for self-sovereign identity, meaning they are fully owned and controlled by the creator and do not rely on centralized third-parties for creation or validation, built on Ethereum" [18]. The uPort technology consists of three main components: smart contracts, developer libraries, and a mobile app. The mobile app holds the user's keys. Ethereum smart contracts form the core of the

identity and contain logic that lets the user recover their identity if their mobile device is lost. Finally, the developer libraries are how third-party app developers would integrate support for uPort into their apps. uPort identities can take many forms: individuals, devices, entities, or institutions.

An identity can be cryptographically linked to off-chain data stores. Each identity is capable of storing the hash of an attributed data blob, whether on IPFS, Azure, AWS, Dropbox, etc., which is where all data associated with that identity is securely stored. Identities are capable of updating this file themselves, such as adding a profile photo or a friend (a peer), or they can also grant others temporary permission to read or write specific files. Since they can interact with blockchains, uPort identities can also control digital bearer assets such as cryptocurrencies or other tokenized assets.

At the core of a uPort identity is the uPort identifier, a 20-byte hexadecimal string that acts as a globally unique, persistent identifier. This identifier is defined as the address of an Ethereum smart contract known as a Proxy contract. The purpose of having a Proxy contract as the core identifier is that it allows the user to replace their private key while maintaining a persistent identifier. If the user's uPort identifier instead were the public key corresponding to their private key, they would lose control over their identifier if they were to lose the device where the private key is held. In the case of device loss, the 'Controller' contract maintains a list of recovery delegates that can help the uPort user recover their identity. These delegates (peers) can be individuals, such as chosen friends and family members, or institutions, like banks and credit unions. A quorum of delegates is required to let the user recover their identity and connect it to a new device.

A problem here is, all recovery delegates of a user are publicly available on the blockchain, which could pose a security risk since an attacker could decide to attack a user's delegates in order to compromise their identity. Additionally, they support holding only a single identity per user, where it is impossible to address the multiple identities that were discussed in the previous chapter.

Sovrin [17] is also a decentralized, global public utility for self-sovereign identity. They claim to offer a lifetime portable identity for any person, organization or thing. There are some dissimilarities in Sovrin compared to uPort.

- uPort is based on Ethereum, a public permission-less ledger for smart contracts (in fact every uPort identity is implemented as at least two and usually three smart contracts). Sovrin is a public permission ledger for self-sovereign identity. So, both the trust model and the incentive model are different.
- On Ethereum, anyone can run a ledger node, so trust in the network is entirely based on the trust in the code (and the coders). With Sovrin, there is a thin layer

of human governance provided by the Sovrin Foundation and the Sovrin Trust Framework, and trusted institutions only operate nodes, so trust is based on both people and code.

- With Ethereum all transactions have an inherent cost. With Sovrin, there is no financial cost to identity transactions; the incentive is to maintain a positive reputation.

2.3 Problem Analysis

Even these kinds of services can change the registered user's device key (personal ID) with the help of predefined delegates (peers) to recover lost identities, and they still have issues like being able to restore identity information by the delegates without the user consent. In other words, once the user has nominated the delegates, the delegates can send identifier restoring commands without a request from the user himself. The user needs not to evoke any command in order to request an identity restoration since the delegates are already been nominated as trusted parties. Therefore, these can lead to possibly tampered keys. A solution for this is proposed in the paper "A Blockchain-based Identity Management Service Supporting Robust Identity Recovery" by Wookun Lee et al. [17].

What they have suggested is an application that stores and manages identity information of users on the Ethereum blockchain platform [18]. It depicts that the users provide lists of their trusted 'friends,' who become peers in the initial registration. The IdM service uses Ethereum smart contracts to store user profile information on the blockchain. If a user loses his or her personal ID, he or she can request the nominated friend to verify their identity on the platform. At this time, the recovery service can ascertain the new personal IDs sent according to the user's intention and considered that the recovery process has not tampered if those IDs are all the same. To use the proposed service in Ethereum distributed applications (DApp) [16], they present the smart contracts required for the service and the relationship between those smart contracts.

That seems like the perfect solution until we think of a situation where the identifiers (identity attributes) change and the initially nominated peer is no longer able to verify the current identity of the user (due to the dynamic nature of an identity). In a situation where the value of an identifier changes, as the user transferred from company X to Y in the previous example, the employer from the company X cannot verify the designation of the user anymore. In such a scenario, the verified details that are already on the blockchain are no longer valid and needs to be re-verified with related peers. In such a case, it might or might not need to nominate new delegates,

according to the scenario.

2.4 Summary

Key concepts related to a blockchain based IdM service are discussed in this chapter.

Commercially available similar solutions, uPort [15] and Sovrin [17] are also introduced, and their limitations are noted. It is explained how and why, although both the IdM systems are well-established and up-and-running in the market, their identity recovering facilities related to dynamic and multiple identities is not up to the expectations when thinking of possible frauds.

This problem is analyzed, and the proposal "A Blockchain-based Identity Management Service Supporting Robust Identity Recovery" by Wookun Lee et al. [17] is supported as a solid foundation for a solution. Even though recovering dynamic identities is not supported in the mentioned proposal, it is recognized as a possible contribution to the domain if the proposal can be optimized accordingly.

In order to design the proposed IdM framework, the foundation is laid through this literature survey.

Chapter 3

Design

This chapter discusses about the design and the methodology for implementing the lost identity recovering feature in a decentralized blockchain-based IdM framework which handles multiple and dynamic identities.

When a user nominates a delegate as a peer, the application will create a smart contract between those two parties. Further explained, all the users of the IdM system are in a decentralized web of trust [19]. These contracts between users and peers will also have a validity period, which can be set up at the initial registration. Even within the validity period, if the related identifier is changed, the smart contract will trigger to get expired and will need to have a new valid contract to verify the new identifier. Until a new contract is formed, the identifier will be classified as "unverified" and the identity related to any unverified identifier is no longer a trusted identity. Any unverified identifier cannot be recovered in a case of identity lost since there is no peer to verify the credentials.

Once a smart contract expires because of any of the above mentioned two reasons, user and a peer (the same one as the previous or a new peer, depends on the context) both have to agree on a new contract. If any of them does not agree to be peers anymore, the contract will not be renewed; hence the identifier is not recoverable. If all the contracts of a single user with the peers are expired, and new contracts have not been made, that account will be removed from the "Recovery Store" [5] dropping them from the decentralized web of trust [19], making the account dormant, since there are no delegates to identify the person as a trusted user anymore. Consequently, the recovery process gets more efficient since the system has to search only among trusted users.

As mentioned earlier, in the journal paper “A Blockchain-based Identity Management Service Supporting Robust Identity Recovery” by Wookun Lee, Jae-Hwan Jin and Myung-Joon Lee [14] will be the base for this project.

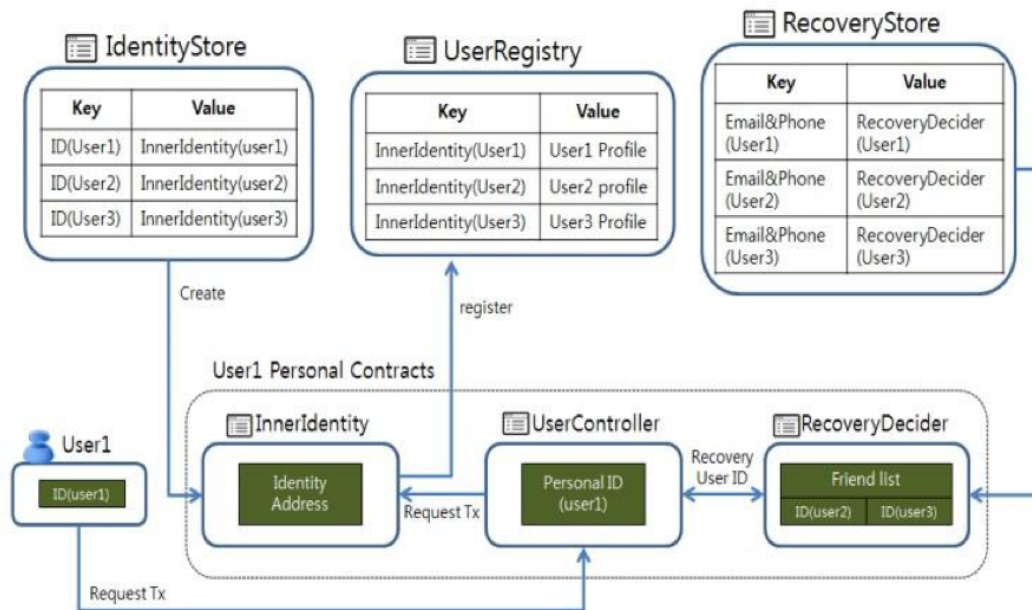


Figure 3.1: Smart contracts for the proposed IDM service [5]

What they suggest is to register users with their information along with a list of trusted friends’ details. The design of the IdM service contains a smart contract named **Identity Store**, which manages the identities of all the registered users. IdM service uses this contract to manage personal smart contracts of each user. Each of these personal contracts is composed of three contracts: **User Controller**, **Inner Identity**, and **Recovery Decider**.

The **User Controller** contract stores the personal ID of the user, which is the key for the user to access the IdM service. The **Inner Identity** contract plays a role of an internal identity which can be used as the user’s real identifier in the service. The **Recovery Decider** contract stores the list of friends for recovery. The address of **Recovery Decider** is stored in the **Recovery Store** contract which maintains identity recovery information for all users registered for the service.

Users can be assigned with their new personal IDs via **Identity Store** at any time. They have to requests for the registration with the service including his or her personal ID and the identifier details along with the peers. Then the **Identity Store** creates personal smart contracts to store profile information (identifiers) and to handle user requests. Once they are created, the user’s recovery information will be maintained in the service for the *user identity recovery service*. The recovery information of a user comprises a list of friends and emergency contact information to be used when

the user loses his or her personal ID. The friend list is stored in the user's **Recovery Decider** contract, whose address is maintained in the **Recovery Store** contract, which has a validity period as mentioned above. At the end of this process, the user can use the IDM service, including the *user identity recovery service*.

The identity recovery is the process of replacing a registered personal ID with a new one so that users of the IdM service can access existing personal contracts even if they have lost the initial Personal ID. According to this design, it is carried out using recovery information submitted during the user registration, which is in the **Recovery Decider** and **Recovery Store**.

3.1 Methodology

Purpose of our research is to maintain a network of trusted people with smart contracts so that if a user lost his/ her identity, they could request the trusted users to verify the identity recovery. These contracts will also have expiry dates and triggers to expire the contracts, making them maintaining up-to-date contracts continuously. Therefore it is certain that the specific users and peers in the IdM service still trust each other. This provides a resolution to the identified issue of handling transforming or dynamic identities. As different contracts can have different triggers to expire, it is also possible to handle the multiple identities by having multiple peers to verify each identifier of a single user.

In order to continue the research and provide a proof of concept, an experimental design will be carried out based on the design science methodology. What we planned to do is designing a working artifact on Ethereum blockchain and evaluate the results to find out whether the proposed solution by W. Lee et al. can be optimized by the solution that we are proposing.

Since the basic knowledge on how smart contracts should act and interact with each other has already been proposed in the above research, those will be implemented initially. After that, if there occurs any issue, they will be refined as demanded. New logic has to be then defined and implemented so that it can be merged with the previously refined implementation. Finally, the work done will be evaluated in order to verify the effectiveness of the proposed solution in recovering an identity recovery in Ethereum based IdM service. There will be two main iterations in the development process since we will be initially implementing the basic solution proposed and then the refined and final proof of concept.

The application for the research project is designed in several steps, as mentioned below.

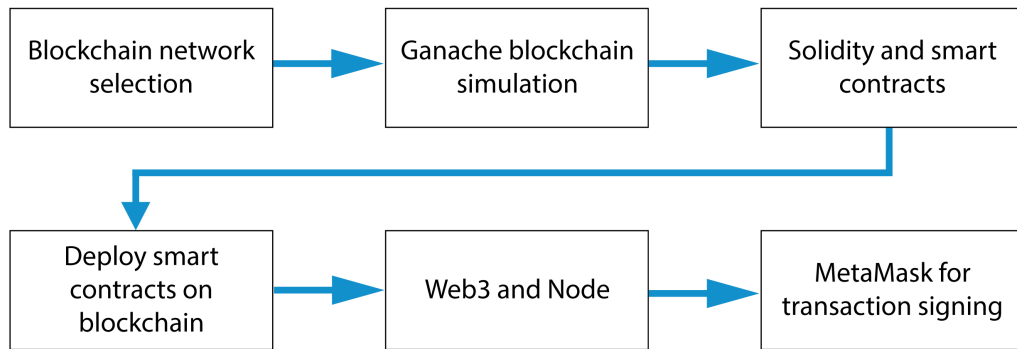


Figure 3.2: Application Design

As the initial step, the suitable blockchain network is selected, which is the Ethereum [18] framework. The base for the project is the selection of blockchain network since the protocols, and the capabilities differ from every available blockchain network.

Ethereum is an altered idea of Nakamoto’s Bitcoin concept [7]. Moving out from the original bitcoin’s blockchain properties, Ethereum decouples the layer of smart contracts from the blockchain protocol. This influences the Ethereum’s capability of creating customized, programmable transactions available digitally (smart contracts [11]), widening the usage towards different online applications.

Once the blockchain network is selected, the next step is to simulate the blockchain with Ganache [20] (discussed further in later chapters). This is an application that allows creating private Ethereum blockchain in order to run tests, execute commands, and inspect various states while controlling its operations. With Ganache, it is possible to perform all actions that would run on real Ethereum network, but without spending any real Ether (i.e., *type of crypto-currency spent in Ethereum transactions*).

With that being done, smart contracts are implemented in order to create the application logic. These are written in solidity [22], which is an object-oriented, high-level language for implementing smart contracts. Smart contracts impose the rules and conditions upon the behavior of the accounts within the Ethereum network.

Smart contracts are deployed on the Ethereum blockchain and tested with Ganache simulator. All the related accounts are monitored for accurate transaction updates and ether balances, minimizing the ether usage in possible approaches.

Node.js [23] and Web3.js [24] are used for the building of the network application. Node.js generates the dynamic page contents within the application from the outcomes of the transactions added on blockchain. This is further supported by

Web3.js, which is a JavaScript library for interacting with The Ethereum blockchain.

Finally, to run the proof-of-concept application on the browser, Metamask [25] is used as the virtual node representing the full Ethereum node. MetaMask is a browser extension that includes a secure identity vault that provides a user interface to manage accounts and sign blockchain transactions.

Chapter 4

Implementation

This chapter includes a detailed elaboration of the implementation process of this project. Here the implementation of the lost identity recovering feature in an IdM service is described along with the tools and technologies used, implementation challenges, remaining issues, and possible solutions.

Since the implementation is carried out in 2 main phases, each phase will be defined separately as follows.

- i. Implementation of the basic IdM service to handle multiple and dynamic identities.
- ii. Implementation of the lost identity recovering methodology.

These are described in the implementation chapter under the following subsections.

1. Development Environment
2. Selection of the tool and libraries
3. Implementation of the basic IdM service
4. Implementation of the lost identity recovering feature
5. Optimization of the IdM framework

4.1 Development Environment

The proof-of-concept application is developed in a machine running MacOS Mojave (Version 10.14.5) with a processor of 1.8 GHz Intel Core i5, and 8GB memory. Chrome (Version 74.0.3729.169) along with MetaMask extension (Version 6.6.1) is used as the browser and associated components for implementation and testing. Ganache (Version 1.2.2) simulated the blockchain in the local environment.

4.2 Selection of the Tool and Libraries

In this section, selection of the tools and libraries is explained and justified.

4.2.1 Truffle Framework

The development environment selected for the application implementation is Truffle framework [21]. Truffle is a testing framework and an asset pipeline based on Ethereum. This consists of its own smart contract compilation methodology which allows effective contract management.

Truffle supports custom contract deployment, library linking, and creating complex Ethereum applications [21]. Truffle permits framework migration; therefore, is useful in creating testing applications. With Javascript files which are responsible for serving uninterrupted contract deployment to the Ethereum network, migration scripts are added as the project evolves. Under the assumption that the deployment needs will change over time, a history of previously run migrations is recorded on-chain through a special Migrations Contract in Truffle. Since Truffle also comes with automated testing tools for contracts in both Javascript and Solidity, an evolutionary development of the DApp [16] is efficient with this framework.

4.2.2 Ganache

Solidity compiler is used in Truffle to compile and deploy smart contracts in the Ethereum network. As the deploying contracts often get changed in the development phase, it is impracticable to deploy these changes to the public blockchain. Everything written to a public blockchain is permanent [9], since not suitable to be done in a developing stage of the DApp.

Ganache [20] is the selected solution to handle this issue. It is an in-memory

virtual blockchain which acts as a blockchain simulator. As it allows temporary contract compilation and migration, Ganache is quite useful at the development stage. This comes in 2 versions, Ganache (with a GUI) and Ganache-CLI. Ganache listens to a port (by default is 8545) which is configurable, and Truffle is needed to be directed to the port so that both are linked in order to work.

Ganache provides 10 virtual accounts with 100 Ether per each account to work with. Along with comes 10 private keys for each account which are used to sign transactions to be written on to the Ethereum blockchain. These are very useful in the sense, as the contracts are being tested, the Ether usage (gas consumption in the blockchain transaction) and the transaction spent is directly shown with the corresponding accounts. This allows the optimization of the DApp as well as in any case of contract migration, a new batch of accounts can be re-generated and tested on further.

4.2.3 Node Package Manager (npm) bundled with node.js

Node.js [23] and the node package manager [26] (npm) is used for package controlling in DApp development. Node is an asynchronous event-driven JavaScript runtime designed to build scalable network applications. The package manager allows adapting available packages for the DApp; therefore, it is easy to use in the development of this research's proof-of-concept application. Several standalone tools are also available for download with npm.

Node.js is identified as a web front-end development tool as well as a cross-development platform. Node executes Javascript codes server-side and often used in developing real-time network applications. As the IdM service has to be spontaneously communicating with the public blockchain, and as the accounts are updating with each and every transaction verified frequently, the event-driven I/O APIs and asynchronous features in Node.js suit the front-end development of the DApp. Moreover, the scalability offered through Node.js is an added advantage in the making of this proof-of-concept application. While in the development phase, it is possible to scale the project vertically by adding more features to each node, and horizontally by adding new nodes to the system in order to test on the performances.

4.2.4 Chrome Browser along with Metamask extension for Google Chrome

With MetaMask, one can run Ethereum DApps from the browser itself, without running a full node.

MetaMask allows sending requests from specified Ethereum nodes, even to the nodes outside the local network [25]. This acts as a secure digital wallet with the ability to sign blockchain transactions. Instead of storing Ether on a local machine, MetaMask stores them in a browser extension. This is convenient because as the security is integrated into the extension, and any node can be accessed with the right credentials from a browser in any machine. Therefore, MetaMask is an excellent tool to use in the development of the DApp.

4.3 Implementation of the Basic IdM Service

As described in the design chapter, the basic IdM design is consisting of the 3 main contracts, *Identity Store*, *User Registry*, and *Recovery Store* and 3 subcontracts, *User Controller*, *Inner Identity*, and *Recovery Decider*. Each of these contracts are written separately in Solidity 4.2.

```
1 pragma solidity ^0.4.2;
2
3 contract IdentityStore{
4     //model a user
5     struct User {
6         uint id;
7         string name;
8         string designation;
9         uint delegateCount;
10    }
11    // store accounts that have delegates
12    //fetch user
13    mapping (address => bool) public verifiers;
14    // store users
15    //fetch user
16    mapping (uint => User) public users;
17    //store user count
18    uint public userCount;
19
20    //verified identifier
21    event verifiedIdentifier (
22        uint indexed _userId
23    );
24    //constructor
25    function IdentityStore () public {
26        addUser("User 1");
27        addUser("User 2");
28    }
29    //add user
30    function addUser (string _name) private {
31        userCount++;
```

```

32     users[userCount] = User(userCount, _name, 0);
33 }
34 function verify (uint _userId) public {
35
36     // need to verify for a valid user identity
37     require (_userId > 0 && _userId <= userCount);
38
39     // record the delegate has verified
40     delegate[msg.sender] = true;
41
42     // update user delegateCount
43     users[_userId].delegateCount++;
44
45     // trigger verifiedIdentifier
46     verifiedIdentifier(_userId);
47 }
48 }

```

Above is the solidity code written for the initial Identity Store contract. In this basic contract, the purpose is to add a new user to the blockchain, set up the user's "designation" identifier, let a delegate to verify the identifier, and increment the delegate count.

Following figure 4.1 shows the corresponding output in the web interface of the IdM application once the designation of the user "TestUser1" at the address "0xCCDFA0be69e410F92dC0Ef042255e7554bBFF1C5" is verified by the delegate at the address "0x1a0457820b3F224F68aa14851fC0A27137E93B74".

User Register

Full Name	TestUser1
Designation	TestDes
Delegate Address	0x1a0457820b3F224F68aa14851fC0A27137E93B74 📋

Delegate Details


Name	TestDelegate1
Valid Until	01/01/2025

CLEAR
REGISTER

Figure 4.1: Web interface for designation verification

The date is a property requested and received from the delegate's side for verification purposes.

User Register

Full Name	TestUser1
Designation	TestDes
Delegate Address	0x1a0457820b3F224F68aa14851fC0A27137E93B74 

Delegate Details



Name	TestDelegate1
Valid Until	01/01/2025

CLEAR REGISTER

Figure 4.2: Web interface for designation verification


In the above figure 4.2, it shows the request received by the delegate at the IdM for verifying the designation of the user. From here, the delegate can accept or reject the verification. If he or she is accepting to verify the identifier, the corresponding expire date has to be set. The "Valid Until" date in the figure 4.1 is this expiration date for the contract between the user and the delegate who verifies the designation identifier. This is set to automatically trigger transactions in order to refute the dynamic identifiers periodically so that the identities are kept verified up-to-date with right identifiers.

User Profile

Full Name	TestUser1
Designation	TestDes 
Gender	Male 

Delegate Details

DESIGNATION ▼

Address	x1a0457820b3F224F68aa14851fC0A 	Valid From	10/04/2019
Name	TestDelegate1	Valid Until	01/01/2025

SHARE KEY

Figure 4.3: User Profile

A single user can register more than one delegate per a single identifier. Above figure 4.3 shows the interface of the IdM's user profile where the delegate details and details of the contracts are displayed.

4.4 Implementation of the Lost Identity Recovering Feature

All the separate identifier validation with each delegate creates separate contracts with related *"Valid Until"* dates, which will automatically be expired on the date. Any identifier without more than a single valid contract among delegates verifying the identifier is removed from the *"Recovery Decider"*, hence from the *"Recovery Store"*. In such a case, there will not be any delegate to recover the particular identifier from the user's identity if the identity has to be ever recovered.

When the user is initially signing up with the IdM service, he or she is given an account address (which is the user's personalID) and the corresponding private key. Ganache simulates this account with a balance of 100 Ethers in the developing environment. The following figure 4.4 shows a successful signing-up to the DApp by TestUser1 from the account *"0xCCDFA0be69e410F92dC0Ef042255e7554bBFF1C5"*.

The screenshot shows the IdM Service Application registration interface. At the top, the title "IdM Service Application" is displayed in blue. Below the title, there is a registration form with two input fields: the first contains "TestUser1" and the second is a password field with dots. Below the form are two buttons: "CLEAR" and "SIGN UP". Below the form is a section titled "Your Credentials" with a blue header. This section contains two rows: "Your Address" with the value "0xCCDFA0be69e410F92dC0Ef042255e7554bBFF1C5" and a copy icon, and "Private Key" with a masked key and an eye icon. Below the "Your Credentials" section is an "EXPORT" button.

Figure 4.4: IdM Service Registration


The application facilitates exporting the address and the key separately to local storage so that the user has access to the exported details if the credentials are forgotten. When using the DApp from the same device that the user signed up, he or she can use the initially provided username and password to log in again. To log in

with another device, address and the private key are required along with the username and password.

Losing either the address or the private key means losing the identity since all the identifiers and their verification are stored in the Ethereum blockchain, and the user has no way to access them without knowing the account details. In order to share the identity to the external, the user must sign the transactions with the private key. Therefore, losing the private key means there is no method of sharing the verified identity with an external party.

In such a case of losing the identity (i.e., losing either the address or private key, or both), when the user logged in by providing the username and password, he or she can request the previously added delegates to re-verify the identifiers one by one. The most important thing to be noted at this stage is that the user will have to re-enter the identifier properties and note the relevant addresses of the delegates as in the following figure 4.5.

Recovery Request

Full Name	TestUser1
Designation	TestDes
Delegate Address	0x1a0457820b3F224F68aa14851fC0A27137E93B74 

Delegate Details

Name	TestDelegate1
------	---------------

Figure 4.5: Request for Identifier Recovery

Therefore it is recommended to copy the relevant addresses to a local storage each time a copy icon is displayed with a new address. There is no risk regarding that, since having only an account address by it's own has no use other than to refer it. In order to use it, one should also have the corresponding private key to sign and authenticate a transaction.

When these requests are sent, they are mapped with the valid contracts in the Recovery Decider. This is a two-way verification before a recovery since each delegate has to accept the recovery request and confirm the given attributes related to each identifier, and the DApp has to identify the valid contracts among two accounts where one account belongs to the delegate that accepts the request.

Suppose that TestUser1 sends a recovery request requesting a verification for TestDes to the address *"0x1a0457820b3F224F68aa14851fC0A27137E93B74"*, the DApp search the blockchain for valid contracts related to the Designation identifier between *"0x1a0457820b3F224F68aa14851fC0A27137E93B74"* and TestUser1. If that delegate accepts the request by signing a transaction verifying the TestUser1's designation is *"TestDes"*, then the DApp can narrow down the contracts found and identify that the exact contract is with the address *"0xCCDEFA0be69e410F92dC0Ef042255e7554bBFF1C5"*. Hence, the DApp can recover the address of the TestUser1.

4.5 Optimization of the IdM Framework

Now that the basic IdM service with the ability to recover a lost identity is implemented, the service is optimized to efficiently recover most of the identifiers and allow the user to continue using the verified identity as usual. As mentioned above, a user can add as many delegates as needed per a single identifier. This not only verifies the identifier better but also increases the efficiency of recovering the identity as there is a higher chance of a delegate accepting a recover request within a short period of time. This reduces the time taken by the DApp to search for all the valid contracts that the delegate have with the users as the pool of users can be reduced.

Chapter 5

Evaluation

Since this is a system developed as a proof-of-concept based on the proposal “A Blockchain-based IdentityManagement Service Supporting Robust Identity Recovery” by Wookun Lee, Jae-Hwan Jin and Myung-Joon Lee [14], it was not possible to implement the IdM service in a real Ethereum Network and let the general public to use. The evaluation for the research is done with a **Focus Group**, where experts and general users from related industries used and experienced the DApp to rate it in various aspects.

The focus group consisted of professionals from the industries where individual identities are highly valued (ex: Banking & Finance, Real Estate, Administration & Governing Bodies, etc.). The main reason behind the selection of focus group evaluation is to obtain detailed information from the industries that have the potential to use the IdM service so that the real users evaluate it. As this domain might be a bit unfamiliar to some of the professionals, a focus group would provide a broader range of results than from individual interviews. Both personal and group perceptions could be gathered with this type of qualitative research technique.

The evaluation is carried out under 3 main aspects.

1. General Knowledge on the concept of “Identity”
2. Usage of available IdM services
3. Comparison of the developed DApp with available IdM service

5.1 General Knowledge on the concept of "Identity"

At the beginning of the discussion, a questionnaire was provided to the participants in order to gather information on their individual knowledge on the concept of "identity".

The questions are as follows.

1. General Knowledge on the Concept of "Identity"

- (a) Do you or your company gather individual's identity related data?
- (b) If yes, what are the 3 most common identifiers you gather?
- (c) How do you verify each identifier?
- (d) Do you feel that the data you are collecting are authentic?
- (e) How do you store identity related data?
- (f) Do you feel that the data you are storing are secure?
- (g) How often do you refer back to the data?
- (h) Do you feel it is easy to refer back to a relevant individual's identity related data?
- (i) If you have lost access to a relevant individual's identity related data, what is the procedure of getting the data back?

The answers for the above questions are analyzed and found the most common answers from 12 individuals from 3 industries.

1. Most Common Answers to the questionnaire for General Knowledge on the concept of "Identity"

- (a) Yes (100%)
- (b) Address, NIC, Monthly Income
- (c) Address- No verification
NIC- True copy
Monthly Income- Individual's income statement/ Individual's salary statement
Date of Birth Birth Certificate
Designation- Employer verification/ Work ID
- (d) Yes (66.66%)
No (8.33%)
Not sure (25%)

- (e) Computer based network
 - Filing cabinet
 - Local computer
- (f) Yes (58.33%)
 - No (0%)
 - Not sure (41.66%)
- (g) Very often (16.66%)
 - Regularly (0%)
 - Rarely (83.33%)
- (h) Yes (41.66%)
 - No (58.33%)
- (i) Request the individual to re-submit the verification documents

From these results, it is concluded that the group has a general idea on what is identity, what are the identifiers that make-up one's identity and their business process of gathering, storing and re-accessing identity related data.

5.2 Usage of Available IdM Services

Then, an introduction to the study was given, and the group was asked to name any IdM tools or services they use in order to keep track of individual identities. Unfortunately, non of the participants were able to name a tool or service.

Therefore to continue with the evaluation, the above discussed uPort[15] and Sovrin [17] were introduced to the group by a presentation. Videos related to their usage was shown and the group is asked to discuss on their advantages and disadvantages compared to their existing techniques of handling identity related data.

This session was finalized by the group with a conclusion that automated IdM services like uPort[15] and Sovrin [17] would definitely increase their efficiency in the businesses since the identities are already verified and they do not have to re-verify the identifiers individually. They agree that many unrelated industries (ex: Banking & Finance vs. Real Estate) can trust individual identities if they are verified by the related industry. For example, the Real Estates agree that they can completely trust the Monthly Income of an individual if the income is already verified by a bank which is a third party in the context.

5.3 Comparison of the Developed DApp with Available IdM Service

Once the advantages and disadvantages were discussed, then the developed DApp is introduced to the group. They were asked to rate the 3 known IdM services (uPort, Sovrin and the developed DApp) according to the below mentioned categories.

3. Comparison of the Developed DApp with Available IdM Service

- (a) Which IdM service do you think is the most effective to use in your industry?
- (b) Which IdM service do you think is having the most user friendly interfaces?
- (c) Which IdM service do you think is the most easy to understood by a new user?
- (d) Which IdM service do you think is capable of recovering lost identity effectively?
- (e) Which IdM service do you think is capable of handling dynamic identities?
- (f) Which IdM service do you think is capable of handling multiple identities?

The below table 5.1 shows the summery of the results found. Out of 12 participants, the number of votes that each IdM service have received is noted here. Highlighted are the applications that have scored the highest in each question.

Q#	uPort	Sovrin	Developed DAapp
a	2/12	0/12	10/12
b	6/12	1/12	5/12
c	3/12	2/12	7/12
d	0/12	1/12	11/12
e	0/12	0/12	12/12
f	7/12	1/12	4/12

Table 5.1: Compression Analysis

This shows that the DApp which we have developed scored the highest votes in questions *a, c, d, and e*, which respectively represents the adaptability to each industries, adaptability of new users to the system, ease of lost identity recovery and handling dynamic identities. Moreover, the DApp has scored the second highest marks in other two questions *b and f*. These are referred to the user interfaces, and handling the multiple identities effectively.

The below table 5.2 shows the scores obtained for each question, and the total score for all the competitive applications. The lowest possible score and the highest possible score is represented by 0 and 3 respectively.

Q#	uPort	Sovrin	Developed DAapp
a	2	0	3
b	3	1	2
c	2	1	3
d	0	2	3
e	0	0	3
f	3	1	2
total score	10	4	16

Table 5.2: Competative Scores

According to the above scoring schema, the developed IdM solution scores the highest of 16 scores. With this results, it can be concluded that the proof-of-concept application developed for the research "Identity Management Service Based on Blockchain and Smart Contracts to Recover Dynamic Identities and Multiple Identities" is a success compared to the two most popular commercially available IdM services, uPort and Sovrin.

Chapter 6

Conclusion

The goal of the research was to develop an IdM service with the ability to recover dynamic and multiple identities. In order to do that, the proof-of-concept IdM application is built. Its performances are evaluated against the two most popular commercially available competitive IdM services. This chapter concludes the overall outcome of the project, and state limitations, possible further investigation, and the future works.

Digital identity is becoming a hot topic from the last few decades, and now it has come to a point where everybody is representing themselves virtually in one another form, by taking their digital identity to existence. With the modern-day technological advancements, keeping one's own digital identity safe and verified is not difficult. There are tons of IdM services up-and-running at the moment, but each having their own specialties and features to suit most of the user needs.

This project is also aimed towards such an IdM solution, with its twist to not only to store digital identities but to recover lost identities in a case of unexpected loss of the identity. Other than just facilitating an identity recovery feature, the solution also expanded to handle dynamic identities and multiple identities too. These two features in real-world identities have not been sufficiently addressed in the available IdM services; therefore this research aimed to expand the focus towards addressing these issues.

6.1 Limitations of the Provided Solution

Even though the project was a success, the main limitation of the research is that the developed IdM solution was designed as a proof-of-concept. It is developed in a way

that it works in an Ethereum based blockchain simulator, but not on the real Ethereum network.

6.2 Future Work

Therefore, the main possible future work is to implement this concept into an IdM service that runs in the Ethereum network. The base design has to be changed by a significant amount in order to do that. The tools that we have used here are selected with the intention of simulating the IdM service and the lost identity recovery process, but when implementing the solution into a real-world application, the tools and technologies might get changed.

Another possible investigation is on verifying the multiple identities that exist simultaneously. The DApp at the moment allows the users to have multiple identities regardless of any condition. However, when thinking of an example situation where an employee at a university sits for an exam within the same university, the employer cannot have a designation related to an examiner for the relevant exam. It is evident that even though there exist multiple identities, there has to be a controlling mechanism to validate the concurrent existence related to the business domain.

6.3 Summery

With that being stated, this project concludes, and paths for further investigation are steered. Since there are vast prospects to think of the blockchain based IdM services, and of other blockchain based services as well, the possible research directions are endless. It is highly recommended to anyone who is interested in the domain to come up with creative solutions to bring down the gap between real-world identities and virtual identities so that we all can digitally represent our true selves in the near future.

References

- [1] J. Suler, "Identity Management in Cyberspace", *Journal of Applied Psychoanalytic Studies*, vol. 4, no. 4, pp. 455-459, 2002.
- [2] A. Jøsang, C. Rosenberger, L. Miralabé, H. Klevjer, K. Varmedal, J. Daveau, K. Husa and P. Taugbøl, "Local user-centric identity management", *Journal of Trust Management*, vol. 2, no. 1, 2015.
- [3] "Identity management", Tec.gov.in, 2018. [Online]. Available: <http://tec.gov.in/pdf/Studypaper/identity%20management%20approved.pdf>. [Accessed: 23-Jun-2018].
- [4] "Identity management (IdM)–an essential knowledge for IT project management", Pmi.org, 2018. [Online]. Available: <https://www.pmi.org/learning/library/identity-management-essential-project-management-8152>. [Accessed: 08-May-2018].
- [5] "Identity management", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Identity_management. [Accessed: 08-Jun-2018].
- [6] J. Elliott, M. Ford and D. Birch, "Managing multiple electronic identities", Enisa.europa.eu, 2018. [Online]. Available: https://www.enisa.europa.eu/publications/mami/at_download/fullReport. [Accessed: 26-May-2018].
- [7] Bitcoin, "Bitcoin: A Peer-to-Peer Electronic Cash System", bitcoin.org, 2017.
- [8] O. Jacobovitz, "Blockchain for Identity Management", The Lynne and William Frankel Center for Computer Science Department of Computer Science, Ben-Gurion University, Beer Sheva, Israel., www.cs.bgu.ac.il, 2018.
- [9] "Blockchain", Blockchain.com, 2018. [Online]. Available: <https://www.blockchain.com/>. [Accessed: 01-May-2018].
- [10] N. Kshetri, "Can Blockchain Strengthen the Internet of Things?", *IT Professional*, vol. 19, no. 4, pp. 68-72, 2017.

- [11] N. Szabo. (1994). Smart Contracts. [Online]. Available: <http://szabo.best.vwh.net/smart.contracts.html>
- [12] N. Szabo. (1997). The Idea of Smart Contracts. [Online]. Available: http://szabo.best.vwh.net/smart_contracts_idea.html
- [13] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things", *IEEE Access*, vol. 4, pp. 2292-2303, 2016.
- [14] W. Lee, J. Jin and M. Lee, "A Blockchain-based Identity Management Service Supporting Robust Identity Recovery", *International Journal of Security Technology for Smart Device*, vol. 4, no. 1, pp. 29- 34, 2017.
- [15] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton and M. Sena, "UPOINT: A PLATFORM FOR SELF- SOVEREIGN IDENTITY", *Blockchainlab.com*, 2018. [Online]. Available: http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT_20161020.pdf. [Accessed: 09- May- 2018].
- [16] "What is a dApp? Decentralized Application on the Blockchain", *BlockchainHub*, 2018. [Online]. Available: <https://blockchainhub.net/decentralized-applications-dapps/>. [Accessed: 11- May- 2018].
- [17] "SovrinTM: A Protocol and Token for Self-Sovereign Identity and Decentralized Trust", *Sovrin.org*, 2018. [Online]. Available: <https://sovrin.org/wp-content/uploads/2018/03/Sovrin-Protocol-and-Token-White-Paper.pdf>. [Accessed: 20- May- 2018].
- [18] "Ethereum Project", *Ethereum.org*, 2018. [Online]. Available: <https://www.ethereum.org/>. [Accessed: 09- May- 2018].
- [19] "Web of trust", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Web_of_trust. [Accessed: 12- May- 2018].
- [20]"Truffle Suite — Ganache", *Truffle Suite*. [Online]. Available: <https://www.trufflesuite.com/ganache>. [Accessed: 06- Dec- 2018].
- [21]"Truffle Suite — Sweet Tools for Smart Contracts", *Truffle Suite*. [Online]. Available: <https://www.trufflesuite.com/>. [Accessed: 06- Dec- 2018].
- [22]"Solidity - Solidity 0.5.3 documentation", *Solidity.readthedocs.io*. [Online]. Available: <https://solidity.readthedocs.io/en/v0.5.3/>. [Accessed: 15- Apr- 2018].
- [23]N. Foundation, "Node.js", *Node.js*. [Online]. Available: <https://nodejs.org/en>. [Accessed: 07- Jun- 2018].

[24]“web3.js - Ethereum JavaScript API - web3.js 1.0.0 documentation”, Web3js.readthedocs.io. [Online]. Available: <https://web3js.readthedocs.io/en/1.0/>. [Accessed: 07- Jun- 2018].

[25]“MetaMask”, Metamask.io. [Online]. Available: <https://metamask.io/>. [Accessed: 09- Jun- 2018].

[26]“npm”, npm. [Online]. Available: <https://www.npmjs.com/package/npm>. [Accessed: 07- Jun- 2018].