Blockchain-based Digital Identity to Reduce Duplicate Identity Verification

P.B.W. Amaradiwakara

Index No : 15000087

Supervisor : Dr. Kasun De Zoysa

Co-supervisor : Dr. M.D.J.S. Goonetillake

February 2020

Submitted in partial fulfillment of the requirements of the
B.Sc in Computer Science Final Year Project (SCS4124)

Protecting Copyright Ownership via Identification of Remastered Music in Radio Broadcasts

P.B.W. Amaradiwakara

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name : P.B.W. Amaradiwakara

.....

Signature of Candidate Date :

This is to certify that this dissertation is based on the work of

Mr. P.B.W. Amaradiwakara

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Supervisor Name : Dr. Kasun De Zoysa

.....

Signature of Supervisor Date :

Co-supervisor Name : Dr. M.D.J.S. Goonetillake

.....

Signature of Co-supervisor Date :

Abstract

Preface

Acknowledgement

Table of Contents

Chapter 1

Introduction

1.1 Background to the Research

Sri Lankan government is still using lots of manual processes for identity verification of people. People have to keep providing their legal documents in order to work with the authority. There are several basic legal documents like “Birth Certificate”, “National Identity Card”, “Driving License”, “Passport” for a person in Sri Lanka. And also have other qualification certificates like “ University Transcript”, “School Leaving Certificate”, “Character Certificate” and more. The current process is, people have to provide their information in a legal manner and claim their assertions. When they need to interact with the authority, they should provide the same information or assertions over again. The problem in this context is people provide their very same information or assertions on different authority sectors and organizations. This is a major problem when considering the government sectors. Since there are millions of people in a country, a government needs a provable claim to identify and verify people.

Official evidence of identity is basic to a person’s capacity to implement their privileges and secure access to a wide scope of essential administrations, for example, medicinal services, training, portable availability, social assurances, and budgetary administrations. Sri Lanka has essential and practical personality frameworks that are very much evolved and hearty. Responsibility for National Identity Card (NIC) is high, revealed at 95 percent for men and 90 percent for ladies, with just little pockets of Sri Lankan culture more averse to claim a NIC. Birth Certificates and National Identity Cards are major identities for verification in Sri Lanka. However, the Birth certificate is a mandatory document for claiming the National Identity Card. But most of the time people need to present both NIC and Birth certificate. When people need to apply for a Driving license or a

university enrollment, again they need to present their birth certificate. This is where the duplication of identity verification happens. Because of this duplication process, people have to submit their verification documents continuously. And it is more costly and time-wasting criteria. The Sri Lankan government produces more than 300,000 birth certificates per year. The birth certificate is the most fundamental identity for identity verification in Sri Lanka.

No.of Births, Deaths and Marriages registered by District, 2017-2018

Districts	2017*			2018*		
	Births	Deaths	Marriages	Births	Deaths	Marriages
Sri lanka	326,052	139,822	169,365	328,112	139,498	170,027

Figure 1.1: Sri Lankan Registration certificates

Birth certificate issued by the Sri Lankan government when birth and every person has it with them. For instance, Pasindu is a civilian in Sri Lanka and has to provide her identity for school applying. Then Pasindu needs to enter a state university and again needs to provide the birth certificate. When Pasindu wants to apply for a driving license, again he needs to present his birth certificate as her identity document. As a solution to this problem, the government can consider a centralized system[1] with a centralized database[2] for identity verification.

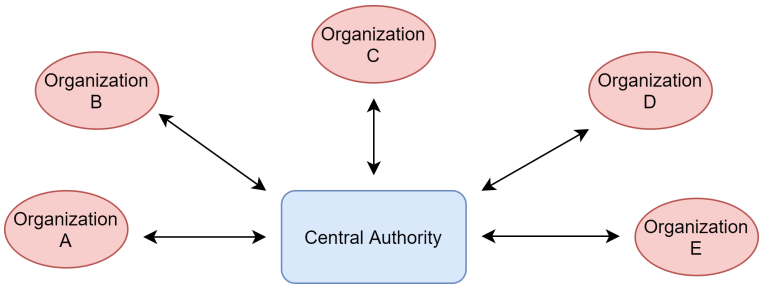

Figure 1.2: centralized database

Centralized systems are systems that use client/server architecture where one or more client nodes are directly connected to a central server. This is the most commonly used type of system in many organizations where the client sends a request to a company server and receives the response. In this centralized system, the government can record the identities of civilians and provide an application program interface (API) for organizations and government sectors that need to verify the identities. Organizations and the government should have a mutual trust for verifying identities. For instance, Alice can verify her claim which could be the birth certificate or any verifiable document with the authority and get an assertion for her claim document. Then Alice can use this assertion when she wants

to apply for a driving license. But this centralized system has some limitations and problems.

1.1.1 Limitations

- It cannot scale up vertically after a certain limit. After this limit, even if it increases the hardware and software capabilities of the server node, the performance will not increase appreciably leading to a cost/benefit ratio ¡ 1.
- Bottlenecks can appear when the traffic spikes – as the server can only have a finite number of open ports to which can listen to connections from client nodes. So, when high traffic occurs like a shopping sale, the server can essentially suffer a Denial-of-Service attack or Distributed Denial-of-Service attack.

1.1.2 Problems

- Highly dependent on the network connectivity – System can fail if the nodes lose connectivity as there is only one central node.
- No graceful degradation of the system – abrupt failure of the entire system
- Less possibility of data backup. If the server node fails and there is no backup, system lose the data straight away
- Difficult server maintenance – There is only one server node and due to availability reasons, it is inefficient and unprofessional to take the server down for maintenance. So, updates have to be done on-the-fly(hot updates) which is difficult and the system could break.
- Only the government can authorize users.

To overcome the above limitations and problems, the authority can think of a decentralized solution[3]. In this decentralized system, the government can distribute the centralized database throughout the organizations that are already trusted by the government.

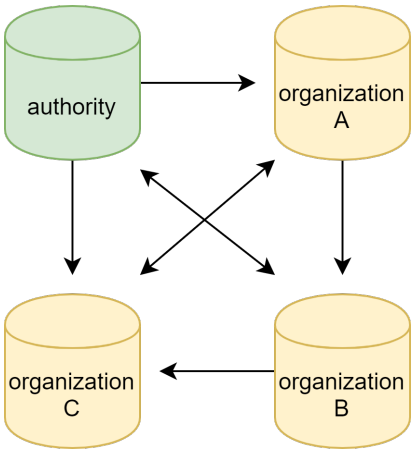

Figure 1.3: decentralized database

The authority can provide privileges for organizations to verify assertions. But still, the government has the ability to provide privileges and there is no transparency. There are some problems in a system like above to archive the following properties.

1. Existence - Users must have an independent existence.
2. Control - Users must control their identities.
3. Access - Users must have access to their own data.
4. Transparency - Systems and algorithms must be transparent.
5. Persistence - Identities must be long-lived.
6. Portability - Information and services about identity must be transportable
7. Interoperability - Identities should be as widely usable as possible.
8. Consent - Users must agree to the use of their identities.
9. Minimization - Disclosure of claims must be minimized.
10. Protection - The rights of users must be protected.

Authorities cannot archive the above properties using a decentralized database solution. The only possible solution to archive those targets is using blockchain[4] architecture. The blockchain network has no central authority and it is the very definition of a democratized system. Since it is a shared and immutable ledger, the information in it is open for anyone and everyone to see. Hence, anything that is built on the blockchain is by its very nature transparent and everyone involved is accountable for their actions. And it is possible to implement the above properties on top of a blockchain model rather than a decentralized database.

1.2 Research Problem and Research Questions

1.2.1 Research Questions

The following questions are founded in this research context.

1. What are the problems of duplicate identity verifications?
2. How to eliminate the duplicate identity verifications?
3. Is it technically feasible to implement a block-chain based identity to reduce verifications in a state?

1.2.2 Objectives

The following are objectives in order to archive solutions for the above research questions.

1. Find out duplicate identity verifications and problems of duplicating identity verifications.
2. Solutions for duplicate identity verifications.
3. Analyzing the feasibility of implementation of the block-chain based identity for a state.

1.2.3 Project Aim

The aim of the research is to utilize computational theories to reduce duplicate identity verifications.

1.3 Methodology

Initially, the current and previous identity management systems have been studied. The first step is to digitalize the current identities and make it reliable for identity owners. After a thorough literature review, an approach is found: self-sovereign identity. And the next objective is to implement it in the real world. and the proposed solution is as follows.

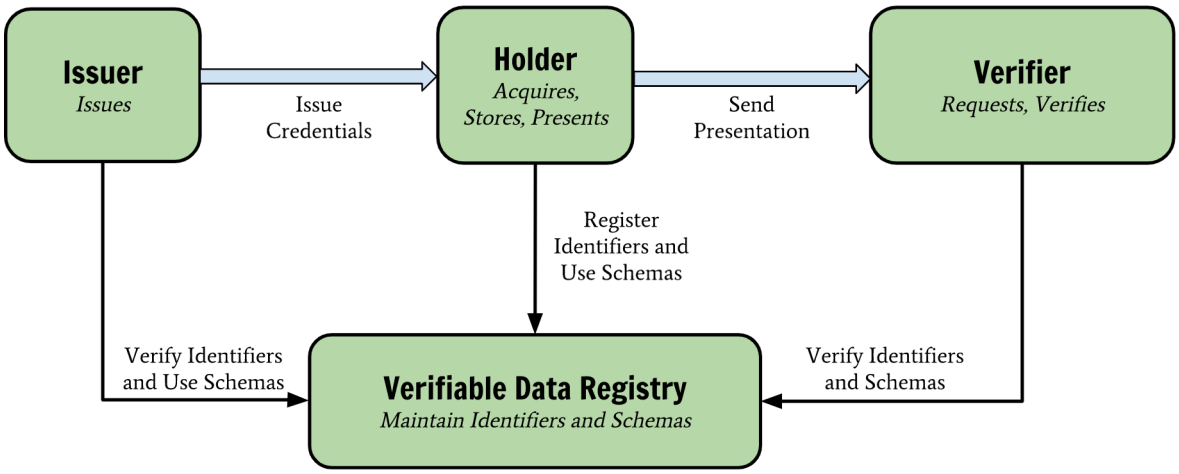

Figure 1.4: methodology architecture

There are 3 main roles in this proposed design.

1. Issuer
2. Holder
3. Verifier

- The holder is the one who wants to verify documents through the system when the holder needs to get credentials for his identity(a legal document or a claim), the issuer should issue the credentials.
- Before issuing the credentials, the issuer needs to contact the government or a trust anchor in the system, to get the schema of the particular identity.
- Then issuer can issue the credentials to the user(holder) and send verify the identifiers using the trust anchor.
- The holder stores the credentials for his particular identity in his wallet.whenever the holder needs to use the credential on a different organization holder sends the credentials.
- Verifier verifies the credentials using the system.

In order to implement this design, blockchain is the most suited technology because blockchain holds the properties of self-sovereign identity and make the system more reliable in between users.

1.4 Scope and Delimitations

1.4.1 In Scope

The following areas will be covered under the research project.

1. Identifying the duplicate identity problems
2. Exploration of possible solutions for duplicate identity verifications.
3. Technical feasibility study of the proposed solution

1.4.2 Out Scope

The following areas will not be covered under the research project.

1. Block-chain smart contract and its functionality.
2. Simulation is running on virtual nodes, not in real distribution systems

Chapter 2

Literature Review

Digital Identity Management is not a replacement discovery within the Information Technology world. Since Ancient Roman times, Passwords have been the ultimate keepers of diversity and security[5]. Even today, they are used to prove one's worth for obtaining some privilege others do not possess, but strongly desire to get. However, over the years with the emergence of the web, Identity Management has transformed through several phases, although maintaining the same goals of,

1. Authentication - Verifying that an individual is who they claim to be.
2. Authorization - Providing them access to some associated resources or services.

These two goals, authentication, and authorization are the backbone of access control[6] and work together to get the specified trust necessary to supply resources or services to assumed unknown persons.

2.1 The Concept and Properties of Digital Identity

The identity concept can be seen from different perspectives and is applicable in different domains, depending on the objective for which digital identity is used. In general, personal identity in philosophy refers to the answer to the question, 'Who am I?' It consists roughly of those properties that make the individual unique and different from others[7]. Precisely, identity refers to a set of qualities and characteristics that make an entity definable, distinguishable, and recognizable compared to other entities[8]. However, in the digital world, "identity" is a set of digital records that represents a user. These records are saved and managed in a standard format by entities that provide the identity information or assurances

needed to complete transactions. Digital identity also accepts and integrates new records to create a rich view of the user[9]. The following are five properties that should be applied to contribute to a more detailed and provisioning solution of a digital identity system.

2.1.1 Entities

According to its definition, an entity is an object that exists or has its own independent existence. Entity conduct as representation from a unit that bears the legal rights for the system, e.g., individuals, businesses, and affiliates. Those entities can be specifically categorized into three types[10]. They have their own private and public keys and can be run by parties that have certain user credentials, such as banks, universities, or other entities that are trusted by the user. The last type consists of Relying parties, which represent a party with which a user intends to interact, essentially, an online service provider; however, in a peer-to-peer system, relying parties can be other users.

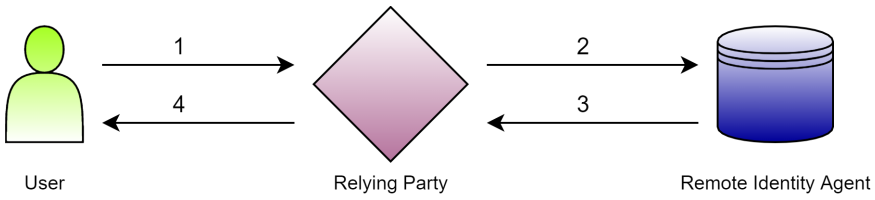

Figure 2.1: Entities

2.1.2 Attribute Type

The attribute type is used to identify the entity. There are three major types[11].

Who you are

In the real world, it is easy to identify a single entity It can include knowledge or data that is only known by that entity, unique physical characteristics of that entity, or items that the entity possesses.

Context

Different constraints on digital identity may be implemented depending on the context. For instance, transferring sensitive information relating to birth certificates over the phone or the internet may be prohibited. level of trust is play a major role here, so the context is used to identify the quantity and type of information needed for providing the trust. For example, in an email context,

the amount of identifying information necessarily is usually only two things: a username and a password.

Profile

After user is verified the profile data is needed for services. User profiles can include what entities can do, what they have subscribed to, what groups they are members of, their selected services, etc. A user's profile can change during the course of an interaction with the service provider.

2.1.3 Lifecycle

There are three fundamental steps to creating digital identity[12]: registration, including enrollment and validation; issuance of documents or credentials; and authentication for service delivery or transactions.

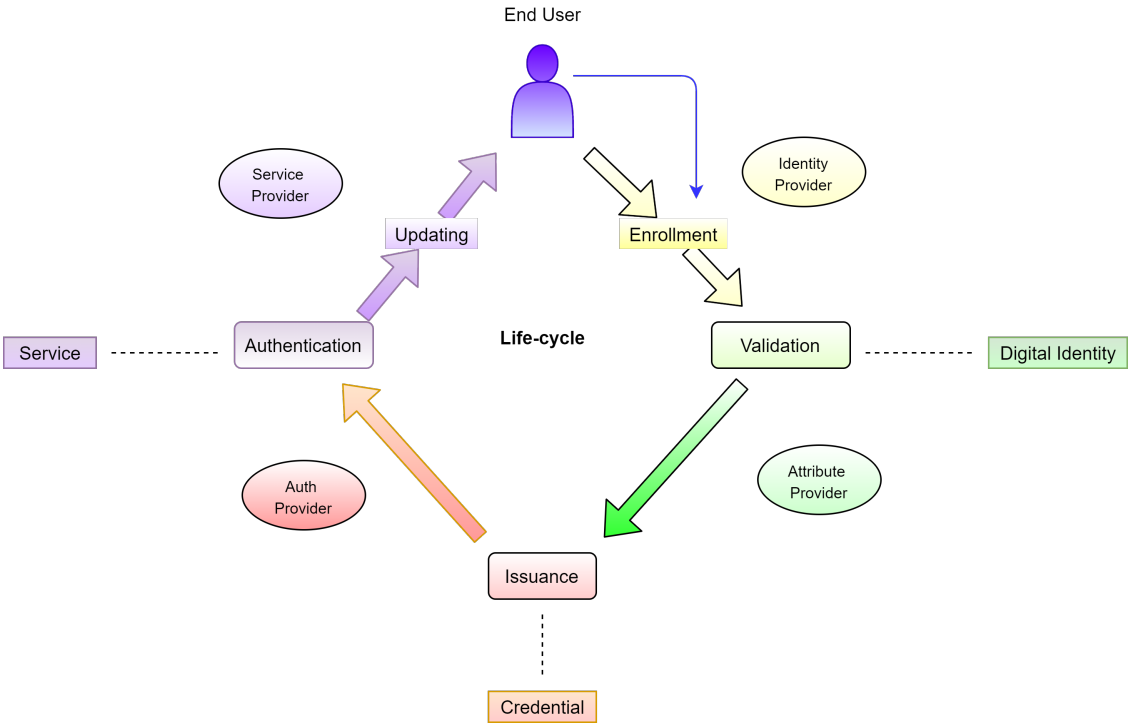

Figure 2.2: Digital Identity Lifecycle

Enrollment

There are two main categories, which are enrollment and validation. Enrollment registration steps: capturing and recording key identity attributes of a person who claims a certain identity. This may include biographical data (e.g., name, date of birth, gender, address, email), biometrics (e.g., fingerprints, iris scan), and the other attributes.

Issuance

Before it can be used by a person, a registered identity goes through an issuance or credentialing process. For an identity to be considered digital, the credentials or certificates (e.g., birth certificate, passport) issued must be electronic, in the sense that they store and communicate data electronically. Types of electronic credentials including smartcards, 2D barcode cards, mobile identity, and identity in the cloud.

2.1.4 Policies

The level of access is conditioned not only by the identity but is also likely constrained by a number of further security considerations, such as the company policy.

2.1.5 Technology

Authentication Techniques

The authentication processors were varying from different factors. the list of major methods are listed below

- Password or personal identification number (PIN)

This advanced method involves hashing and the techniques and data are then exchanged with the server and stored [10]. The PIN technique basically has the same mechanism as a password, but it consists of a numeric term only (usually with four to six digits). still, in the world, the pin-based process is following in banks and other business industries.

- Token

This works using the two-factor authentication (2FA) principle. Instead of using a username and password, a level is added on to obtain a time-limited token (typically a cryptographic key or password) that is used for further transactions during the session. The physical device for tokens mostly does not require an internet connection because it communicates using mobile telecommunication service operator services such as voice calls, SMS, or USSD.

- Public key cryptography

This method utilizes cryptographic mechanisms that, as their underlying theory, engage an asymmetric key pair: a public key and a private key [13]. using the key

pair, data encryption and decryption can be done as well as the signing the data can be done using these key pairs.

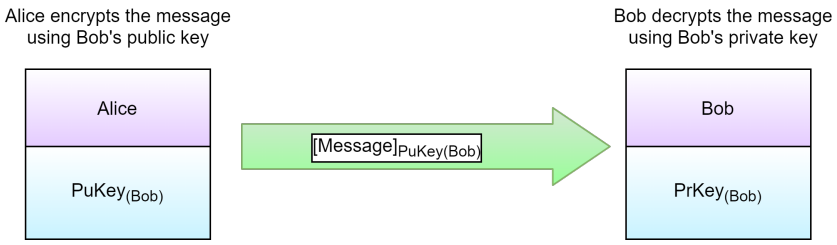

Figure 2.3: Keypair cryptography

- Biometric

Biometric authentication requires a completely different style of the authentication process. This approach is based on a person's biological uniqueness and it can be used for the biometric identification of a person [14], using, for example, fingerprint or iris recognition.most of the pattern matching techniques used for this. Biometrics also depends on the devices that are collect data.

- Smart Card

When used for logical access, smart card technology typically comes in two forms: a credit-card-sized plastic card or a USB device, each with an embedded computer chip [15]. the major application of a smart card is storing a password.

Security Protocols

security protocols highly valued because verification and authentications are depended on that. The widespread authentication protocols used to address security issues within open networks are Secure Sockets Layer (SSL), IP Sec, Secure Shell (SSH), and Kerberos [16].

Storage

There are two new technologies that may offer improved methods in database storage [9]. The first is distributed ledger technology or blockchain combined with encryption and cloud storage, and this allows information to be held and transferred point-to-point in a dispersed, immutable network. The second consists of federated identity standards, such as SAML 2.0, which create interoperability between identity management networks and external applications, allowing federated identity systems to scale to accommodate large numbers of identity providers and relying parties.

single-sign-on (SSO) authentication. plays a major role in the authentication context in this era. This system allows users to sign on only once and have their identities automatically verified by each application or service they want to access afterward[17]. This system serves different purposes[18]. It communicates between applications and the network, it enables communication to applications that are connected by the internet using web resources, and it integrates different domains with different sets of credentials located all over the world. The aim of using SSO is to improve the communication and security of user authentication and access permission verification and also to decrease management costs [19]. Popular examples of SSO systems are found at Google, Microsoft, Facebook, and Yahoo; they provide SSO to their users when accessing emails, groups, documents, and other facilities embedded in their SSO system

Chapter 3

Design

Chapter 4

Implementation

This chapter elaborates the implementation details of the proposed design.

4.1 Indy framework

Indy is a subproject of hyperledger project. It is designed and developed by the Linux Foundation. Hyperledger project has several sub-projects for different use cases. Indy is developed for the Identity Management use case. For all these reasons, the Indy Project has developed specifications, terminology, and design patterns for decentralized identity along with an implementation of these concepts that can be leveraged and consumed both inside and outside the Hyperledger Project. This framework allows to build a normal blockchain(a public blockchain). The research design should be implemented on top of that. In order to that, the first step is to dockerize the environment. The following implementations are going on top of that a docker image which has a couple of nodes as docker containers.

The implementation is running on Jupyter notebook as python environment and codebase is written in python.

4.2 Configure the pool

Nodes are located in the node pool which is the network. Every node has a different IP address and networking every one of them makes the node pool. Those nodes are stored inside the pool ledger(node transactions). This is the place where blockchain sets up its genesis transaction path.

```
await pool.set_protocol_version(2)

pool_ = {
```

```
    'name': 'pool1',
    'config': json.dumps({"genesis_txn": '/home/indy/sandbox/pool_transactio
}
print("Open Pool Ledger: {}".format(pool_['name']))

try:
    await pool.create_pool_ledger_config(pool_['name'], pool_['config'])
except IndyError as ex:
    if ex.error_code == ErrorCode.PoolLedgerConfigAlreadyExistsError:
        pass
pool_['handle'] = await pool.open_pool_ledger(pool_['name'], None)
```

4.3 Setting up Steward Role

Steward is the one who created the chain and genesis block are defined by this role. It creates the NYM transaction(creation of a DID in the ledger) and created a wallet. Indy has this concept called a wallet that is secure storage for DIDs, keys and other crypto materials. The steward's wallet is created and store the did. Then steward can obtain trust anchor role.

```
steward = {
    'name': "Sovrin Steward",
    'wallet_config': json.dumps({'id': 'sovrin_steward_wallet'}),
    'wallet_credentials': json.dumps({'key': 'steward_wallet_key'}),
    'pool': pool_['handle'],
    'seed': '000000000000000000000000Steward1'
}

await create_wallet(steward)

print("\"Sovrin Steward\" -> Create and store in Wallet DID from seed")
steward['did_info'] = json.dumps({'seed': steward['seed']})
steward['did'], steward['key'] = await did.create_and_store_my_did(steward['
```

These DIDs are using base58Check encoding.

4.4 Obtaining verinym for organizations

Every identity owner has a legal identity. A verinym is a DID which related to this legal identity. In order to join the chain organizations should create this verinym. It happens in a couple of steps.

1. Creates a wallet if the organization is new to the chain
   Birth certificates are issued by the Registrar General's Department(RGD). This RGD would be the first organization that register with this chain.

```
await wallet.create_wallet(rgd['wallet_config'], rgd['wallet_creden
rgd['wallet'] = await wallet.open_wallet(rgd['wallet_config'], rgd[
```

2. Create a DID in the wallet.

```
# RGD Agent
(rgd['did'], rgd['key']) = await did.create_and_store_my_did(rgd['w
```

3. Registrar General's Department creates a message with DID and key.

```
# RGD Agent
rgd['did_info'] = json.dumps({
    'did': rgd['did'],
    'verkey': rgd['key']
})
```

4. Send the DID info message to Steward. This is how a communication channel is created.

```
steward['rgd_info'] = rgd['did_info']
```

5. Steward is a trust anchor, so he can send the transaction to the ledger(NYM transaction). But the DID owner is RDG.

```
# Steward Agent
nym_request = await ledger.build_nym_request(steward['did'], stewar
                                             steward['rgd_info']['ve
await ledger.sign_and_submit_request(steward['pool'], steward['wall
```

After this step, RGD has a DID from his identity in the ledger. The every other organization should do the same in order to create a DID. The government is also an organization registered in the chain.

4.5 Schemas Setup for credentials

Schemas are structures that consist of attributes names and details of a credential. Credentials cannot be altered, that is why it maintains a version number. The government is a trust anchor which already created. Every trust anchor can publish credential schema to the ledger.

Schema for Birth certificate

```
{
  'name': 'Birth Certificate',
  'version': '1.0',
  'attributes': ['first_name', 'last_name', '', 'birth_date', 'sex', 'Addres
}
```

1. Create the schema

```
# Government Agent
birth_certificate = {
    'name': 'Birth Certificate',
    'version': '1.0',
    'attributes': ['first_name', 'last_name', '',
                'birth_date', 'sex', 'Address, 'Registrar's_devisio
}

(government['birth_certificate_schema_id'], government['birth_certi
    await anoncreds.issuer_create_schema(government['did'], birth_c
                                         json.dumps(birth_certificat
birth_certificate_schema_id = government['birth_certificate_schema_
```

2. The government (trust anchor) sends the schema to the ledger.

```
# Government Agent
schema_request = await ledger.build_schema_request(government['did'
await ledger.sign_and_submit_request(government['pool'], government
```

4.6 Setup of credential definition

Issuer signs the credentials here.

1. RDA gets the credential schema.

```
# RGD Agent
get_schema_request = await ledger.build_get_schema_request(rgd['did
get_schema_response = await ledger.submit_request(rgd['pool'], get_
rgd['birth_certificate_schema_id'], rgd['birth_certificate_schema']
```

2. Create a credential definition. Then it stores in the wallet of RGD(cannot read)

```
# RGD Agent
birth_certificate_cred_def = {
    'tag': 'TAG1',
    'type': 'CL',
    'config': {"support_revocation": False}
}
(rgd['birth_certificate_cred_def_id'], rgd['birth_certificate_cred_
    await anoncreds.issuer_create_and_store_credential_def(rgd['wal
                                                           rgd['birth_
                                                           birth_certi
                                                           json.dumps(
```

3. Update the ledger by sending the credential definition. It is done by the trust anchor(RGD).

```
# RGD Agent
cred_def_request = await ledger.build_cred_def_request(rgd['did'],
await ledger.sign_and_submit_request(rgd['pool'], rgd['wallet'], rg
```

4.7 The user gets a credential (a birth certificate)

There is a user with no credentials initially. This is how the user gets credentials. A user called "Pasindu" gets a birth certificate from the Registrar General's Department.

```
# RGD Agent
rgd['birth_certificate_cred_offer'] = await anoncreds.issuer_create_credenti

pasindu['birth_certificate_cred_offer'] = rgd['birth_certificate_cred_offer'
```

This birth certificate is issued by the RGD. Then the user needs a master key to the wallet

```
# Pasindu Agent
pasindu['master_secret_id'] = await anoncreds.prover_create_master_secret(pasi
```

In order to make a credential request, the user needs to credential definition.

```
# Pasindu Agent
get_cred_def_request = await ledger.build_get_cred_def_request(pasindu['did_
get_cred_def_response = await ledger.submit_request(pasindu['pool'], get_cre
pasindu['birth_certificate_cred_def'] = await ledger.parse_get_cred_def_resp
```

Now, the user can request the credentials

```
# Pasindu Agent
(pasindu['birth_certificate_cred_request'], pasindu['birth_certificate_cred_
    await anoncreds.prover_create_credential_req(pasindu['wallet'], pasindu[
                                                    pasindu['birth_certificate_

rgd['birth_certificate_cred_request'] = pasindu['birth_certificate_cred_requ
```

The organization(RGD) creates the schema with raw and encoded values.

```
# RGD Agent
# note that encoding is not standardized by Indy except that 32-bit integers
birth_certificate_cred_values = json.dumps({
    "first_name": {"raw": "Pasindu", "encoded": "113948171645748869017221791
    "last_name": {"raw": "Lakshitha", "encoded": "532164278024179012358790 24
    "birth_date": {"raw": "1995.09.25", "encoded": "12434523576212321"},
    "sex": {"raw": "male", "encoded": "2213454313412354"},
    "Address": {"raw": "76,Biyagama RD,Kelaniya", "encoded": "31241412314225
    "Registrar's_devision": {"raw": "Gampaha", "encoded": "2015"},
    "father_name": {"raw": "K.A. Perera", "encoded": "2414123142254354"},
"mother_name": {"raw": "M.L. Wickramsinghe", "encoded": "14123142"},
```

```
    "certificate_number": {"raw": "231131", "encoded": "4545454313412354"},


    })
    rgd['birth_certificate_cred'], _, _ = \
        await anoncreds.issuer_create_credential(rgd['wallet'], rgd['birth_certi
                                                 birth_certificate_cred_values, N

    pasindu['birth_certificate_cred'] = rgd['birth_certificate_cred']
```

Now, credentials for a birth certificate(credential) has been issued by the RGD(issuer/organization). and Pasindu(User) can store the credentials in his wallet.

```
    # Pasindu Agent
    await anoncreds.prover_store_credential(pasindu['wallet'], None, rgd['birth_
                                            pasindu['birth_certificate_cred'], p
```

4.8 How to use the credentials

Now user has his credentials stores in the wallet. There is another organization that is in the chain called the Department of Motor Traffic(DMT/RMV) like RGD. after established the connection between the user(Pasindu) and the organization(RMV), the user got the license application proof request in order to apply for a driving license.

```
    # RMV Agent
    rmv['application_proof_request'] = json.dumps({
        'nonce': '1432422343242122312411212',
        'name': 'Driving-license-Application',
        'version': '0.1',
        'requested_attributes': {
            'attr1_referent': {
                'name': 'first_name'
            },
            'attr2_referent': {
                'name': 'last_name'
            },
            'attr3_referent': {
```

```
            'name': 'sex',
            'restrictions': [{'cred_def_id': rgd['transcript_cred_def_id']}]
        },
        'attr4_referent': {
            'name': 'birth_date',
            'restrictions': [{'cred_def_id': rgd['transcript_cred_def_id']}]
        },
        'attr5_referent': {
            'name': 'address',
            'restrictions': [{'cred_def_id': rgd['transcript_cred_def_id']}]
        },
        'attr6_referent': {
            'name': 'phone_number'
        }
    }
        })

pasindu['application_proof_request'] = rmv['birth_certificate_cred']
```

The proof request asks to sex, address, birth_date in the credentials should be asserted by an issuer and the key of the schema. Only several attributes can be verifiable in the above proof request.

The user(Pasindu) has only one credential to prove the requirements in his wallet. In order to fill the application, Pasindu can decide what attributes should be revealed.

```
#Pasindu Agent
pasindu['application_requested_creds'] = json.dumps({
    'self_attested_attributes': {
        'attr1_referent': 'Pasindu',
        'attr2_referent': 'Lakshitha',
        'attr6_referent': '123-45-6789'
    },
    'requested_attributes': {
        'attr3_referent': {'cred_id': cred_for_attr3['referent'], 'revealed'
        'attr4_referent': {'cred_id': cred_for_attr4['referent'], 'revealed'
        'attr5_referent': {'cred_id': cred_for_attr5['referent'], 'revealed'
    },
    'requested_predicates': {'predicate1_referent': {'cred_id': cred_for_pre
```

```
})
```

The user needs to reveal only 3 attributes(sex,birth_date, address). then the user can make a proof request for the application.

```
# Pasindu Agent
pasindu['apply_license_proof'] = \
        await anoncreds.prover_create_proof(pasindu['wallet'], pasindu['appl
                                                pasindu['master_secret_id'], pas

rmv['apply_job_proof'] = pasindu[apply_license_proof]
```

The proof that RMV received is as follows

```
# RMV Agent
{
    'requested_proof': {
        'revealed_attrs': {
            'attr4_referent': {'sub_proof_index': 0, 'raw':'birth_date', 'en
            'attr5_referent': ['sub_proof_index': 0, 'raw':'sex', 'encoded':
            'attr3_referent': ['sub_proof_index': 0, 'raw':'address', 'encod
        },
        'self_attested_attrs': {
            'attr1_referent': 'Pasindu',
            'attr2_referent': 'Lakshitha',
            'attr6_referent': '123-45-6789'
        },
        'unrevealed_attrs': {},
        'predicates': {
            'predicate1_referent': {'sub_proof_index': 0}
        }
    },
    'proof' : [] # Validity Proof that RMV can check
    'identifiers' : [ # Identifiers of credentials were used for Proof build
        {
            'schema_id': certificate_schema_id,
            'cred_def_id': rgd_birth_certificate_cred_def_id,
            'rev_reg_id': None,
            'timestamp': None
```

```
            }
        }
    }
```

Rmv got all the required details and how it verifies as follows.

```
    # RMV Agent
assert await anoncreds.verifier_verify_proof(rmv['application_proof_request'], r
```

As the final outcome the organization(RMV) can accept the client(register for the driving license).

```
    # RMV Agent
    rmv['certificate_cred_offer'] = await anoncreds.issuer_create_credential_off

    pasindu['certificate_cred_offer'] = rmv['certificate_cred_offer']
```

Chapter 5

Results and Evaluation

Chapter 6

Conclusions

References