Eötvös Loránd University
Faculty of Informatics

SECURE IDENTITY MANAGEMENT ON THE BLOCKCHAIN

Adám Nágy
Assistant Professor at ELTE

Andreas Peter
Assistant Professor / Study Coordinator – Cyber Security & Safety at University Of Twente

Zoltán Hattyasy
Senior PM & Architect Mission Critical Systems at E-GROUP

Kwadjo Anobaah Nyante
Msc. Security & Privacy
Advanced Cryptography

Budapest, 2018.

Summary

One major topical issue that has generated a lot of controversy in the cyber security landscape is Secure Identity Management. So serious is this issue that many educationists and academicians have expressed varying concerns, proposals, and solutions about the subject.

Traditional Identity Management solutions delegate trusted centralized organizations / multiple centralized agencies (service providers) and task them with securely storing the private data of users and providing these users with identity tokens such as ID cards, certificates, login credentials, hardware, and passports. With these identity tokens, users can uniquely access resources and services from the respective service providers. This approach has resulted in four main classes of problems namely: Individual user problems, Information Sharing problems, Governmental information coordination problems, and Privacy problems.

These problems are particular conspicuous in the banking sector when it comes to Know-Your-Customer Processes (KYC). It is expensive and time consuming to do the necessary background checks on customers and their transactions for compliance agencies. At the same time, these background checks have create severe privacy issues that need to be addressed.

In this research, Distributed Ledger Technologies (blockchain) are used to solve these major problems. A hybrid solution is proposed, which is a combination of:

1. A blockchain Gateway Solution, which supports legal compliance and traditional Identity Management features that require strong authentication. This solution serves as a trust anchor that securely links Identity Data to the blockchain and is based on research of the EU's Identity Network eIDAS.[1]

2. A general blockchain Identity Framework, which serves as the fabric for maintaining, verifying and performing transactions using decentralized identities.

Together these two solutions provide a regulatable pseudonymous identity framework that can be used to solve various real world problems. Generalized formal definitions for an

[1] eIDAS is responsible for providing electronic Identification assurance for digital services provided within the EU and its member states

Identity Management Scheme, Self-Sovereign Identity Management Scheme, Claims,Trust, as well as the necessary supporting cryptographic proofs have also been proposed to serve as the foundation for both solutions. In the same light, 3 broad configurations have been proposed to govern and generalize how identity and trust are conceptualized.

Identity Management Systems in general come with trade-offs between legal compliance (usability) and security and user privacy (anonymity). Conventional digital applications provide high legal compliance due to the control by service providers. These come at the cost of lower user privacy and security. Conversely, Distributed Ledger Technologies (blockchain) provide high anonymity / privacy and security at the cost of legal compliance and usability.

Consequently, it is crystal clear that, on one end of the spectrum, conventional Identity Management Systems could greatly benefit from extending their use cases to the blockchain. On the other end of the spectrum, blockchain provides high security and privacy/anonymity, however, in some cases where strong authentication is required, it is also beneficial to extend the blockchain to the real world.

Therefore, this research explores the thin line between the blockchain (Decentralized) and conventional (Centralized) Identity Management Systems and attempts to establish a middle ground between these two fields, thus generating trust on the blockchain with decentralized identities. This middle ground constitutes a technological advancement in both fields (blockchain and Identity Management Systems), and provides the foundation for achieving the delicate balance between security, privacy, and legal compliance (example GDPR).

Acknowledgement

The dream to change the world, especially cyber space, by making it a better and safer place is one that has been shared by most if not all people in the Information Technology discipline. It gives me great joy to say that my master thesis has brought me one step closer to achieving this dream.

With my deepest and sincerest gratitude to God for blessing me with vision and strength to see this through, I also say a very big "thank you" to my supervisors: Ádám Nagy, Assistant Professor at Eötvös Loránd University (ELTE – IK), Andreas Peter, Professor at the University of Twente(SCS) and Zoltán Hattyasy, Senior PM & Architect (Mission Critical Systems) at E-GROUP, Budapest(Hungary). Without their clear and continuous guidance, this research would have been a far cry from a complete success.

My appreciation also goes to the various coordinators of this program, EIT Digital, and all the hardworking men and women there who toil day by day to make things like this possible, the warm staff at E-GROUP and the Center for Cyber Safety and Education in Florida.

Last but definitely not least, I say "I appreciate it" to my family, friends and especially to my amazing and supportive partner Masha Bekjkovikj, who have all kept me grounded and focused during this period.

To all of you and everyone that supported directly or indirectly, it should be known that I couldn't have done it without you.

It is my sincere hope that this research and its results go a long way to make the entire cyber space safer and more secure while simultaneously guarding the privacy of individual users and organizations.

Table of Contents

Table of Figures

List of Tables

Chapter 1

Introduction

An Identity Management System is a branch of Information Technology that governs how digital technologies can be used for enterprise or cross network Identity Management [MW].

Identity Management on the other hand is a branch of cyber security that describes the management of individual identities, their authentication, authorization, roles and privileges within or across system and enterprise boundaries with the goal of increasing security and productivity while reducing cost, downtime and repetitive tasks [ALC+98]. Identity and Access Management (IAM) initiatives have traditionally required multi-year implementations before delivering full value. However, in recent state of the art systems, lengthy IAM deployments are unacceptable, thus, the trend has shifted to more agile IAM technologies that deliver business value within 12 months or less. [Gar]

Typically, these include Office 365 IAM, the Rise of third wave Multi-Factor Authentication (MFA), Infrastructure as a Service (IaaS) for application migration, Single Sign On (SSO), and many more [Gar]. In recent times, there have also been quite a number of Decentralized Identity Management Systems such as: Microsoft ID2020 [id2], IBM Identity Management with Securekey Technologies [ibm] , Estonian Citizenship Identity [Ped13], Sovrin [KRGP18], IdentityChain [DT], ERC Identity #725 [Vogb], among others.

The main goal of these Identity Management Systems can be summarized as establishing trust amongst entities that share a mutual distrust at time of contact. However, these solutions have either been centralized (causing the identity provider to perform several roles such as storage of sensitive information, authentication, and authorization, and hence, increasing the risk of central data silos as well as reducing user control and privacy over data), or weakly decentralized, thus, not providing full user control / privacy and / or legal compliance.

How does one achieve a secure, industrially functional decentralized identity which can also adhere to compliance (example GDPR [Conb]) if needed? This is the main issue explored by this research because the realization of a compliant self-sovereign identity has the

potential for increasing the general security and privacy as well as substantially reducing the cost of IAMs. A point in case is the potential for Identity Management in the Finance Industry. Banks (online and offline), financial institutions, and Fintech companies can reduce KYC costs and other onboarding costs while simultaneously increasing security, user privacy and control. [Yoh]

In the light of the above, the main objective of this thesis is to establish a secure middle ground between centralized and decentralized identities thus, achieving the best of both worlds. To realize this goal the following questions need to be answered:

1. How do we securely store, update and retrieve trusted identity data from the BlockChain?
2. What are the minimum security requirements needed to enforce secure granular data management on the BlockChain?
3. To what extent is the chosen method in 1 above scalable?
4. Are there any other use cases for such a solution in 1 above?
5. How do we achieve compliance in use cases that require strong authentication (Example Governmental level, Banking)?

1.1 Target Group

This research is geared towards generating trust efficiently and securely amongst entities with no prior information about one another. Therefore, it may serve as a blueprint for organizations looking to extend their traditional services to use Distributed Ledger Technologies (BlockChain). For example Banks may choose this research to evaluate their BlockChain solutions for cutting costs in Know-Your-Customer (KYC) processes, thus fighting money-laundering.

It may also serve as a guide to provide the foundation for the transition from Centralized Identity Management Systems to more Decentralized Identity Management Systems, specifically, Self- Sovereign Identity Management Systems. Therefore, students, teachers, and stakeholders of Identity Management schemes may find it useful to understand the transition between these systems as well as the pros and cons of various choices for Identity Management.

The result section of this research may also serve as a whitepaper for the implementation of a self-sovereign Identity Management system while simultaneously filling the research gap involving the necessary technical implementation details necessary to achieve GDPR compliance.

Lastly b Enthusiasts, Organizations looking to deploy blockChain solutions, Cryptography lovers, DAPP programmers, as well as privacy enthusiasts may also use this research

to develop a better understanding of technological Identity and Access Management solutions and how to develop them.

1.2 Personal motivation

At a young age, about 12 years old, my aunt sent me my first computer. It was a second-hand password-protected Windows computer, and truth be told computers were a bit of a luxury in Ghana at the time. My curiosity got the better of me and after a week of research, I broke the password using Rainbow Tables.

To my parents and those around me, I always had a particular proclivity for tinkering with electronics and most times eventually destroying them. However, as my fascination and passion for "loopholes" (I now call it cyber security) grew, it turned out that people actually pay to get their computer systems tinkered with, therefore, I pursued a career in Computer Engineering.

I am currently pursuing a double Master program at EIT Digital Master School in Cyber Security and Privacy and as I drew closer to the end of this 2-year period, specializing in Advanced Cryptography, my interest grew, considerably, in topics such as Privacy Enhancing Technologies and Distributed Ledger Technologies (BlockChain).

A few months ago, the proverbial falling apple hit me on the head, bringing me to the realization that Identity, Privacy and BlockChain are a match made in heaven. This realization, I believe, is partly because I grew up in a part of the world where people are not able to fully enjoy some of their basic human rights and privileges (insurance, medical care, travel, and more), simply because they cannot prove their identity.

In general terms, I feel strongly about Article 6 of the Universal Declaration of Human Rights. "Everyone has the right to recognition everywhere as a person before the law". However, about 1.2 billion people in the world live without documental proof of their existence. The other 6.4 billion have to fanatically guard these documents (passports, citizenship cards, insurance cards, student cards, school diplomas, and more) in order to avoid severe complications.

In the light of the above, it is crystal clear that the world will be much better and hopefully safer if identifying oneself was made much simpler, privacy preserving and more secure. It is my wish that this research brings us all one step closer to this dream.

1.3 Research method

This research considered two general research methodologies namely, Quantitative and Qualitative methods. Quantitative methods are generally more suited to researches that

collect quantitative data in order to verify a hypothesis or question them. If our research was on comparing quantitative metrics of various Identity Management Systems of verify some comparative hypothesis about IAMs, this method would have been more suitable.

However, the nature of this 5 research questions as shown in the introduction drive us to deeply understand the meaning of Identity Management Systems and improve the technology. The expected results here are strictly intangible, thus our choice to use Qualitative research methods.

The research progressed through 4 major phases (some concurrent) to arrive at the results.

1. **Preparation Phase**: In this phase, various Identity Management Systems as well as BlockChain solutions were thoroughly researched in order to find their relative strengths, and weaknesses. This phase was not strictly practical. In some cases like the BlockChain (Ethereum & Bitcoin), and ERC Identity #725, test accounts are created and some Smart contracts are deployed to better understand the technologies.

2. **Design phase**: In this phase, a design for a solution is established, which aims to include all the strengths of the research solutions in phase 1 as well as remove the persistent weaknesses. Thus an optimal solution.

3. **Expert Verification phase**: Interviews, in person and over Skype, were conducted with the leading experts in the field in order verify the design solution provide in phase 2. These interviews were conducted with E-Group in Budapest (E-Group has an existing identity management product line and is developing the Hungarian eIDAS node, which interfaces with the European Union's IAM), OTP Bank, Posta Italiane, TU Berlin (team responsible for the design of Identity Chain (now Digital Identity Management System - DIMS)), and many more.

4. **Cryptographic Verification phase**: Once the solution from phase 2 is verified at phase 3, I also verify that it makes cryptographic / mathematical sense. In this phase formal definitions as well as cryptographic proofs of security are provided the buttress the results of phase 3.

Each phase was carefully documented and eventually distilled into the writing.

1.4 Structure of this report

Chapter 1 introduces the research, highlighting the field of this research, recent state of the art in this field, the current problems and the benefits of a solution (hence benefits of the research). It also covers the target Group for this research, research methods used, as well as my personal motivations for embarking on this research.

Chapter 2 covers the necessary literature required to understand this research. These include Identity management systems, Economics of security (including some game theory), Distributed Ledger Technologies (BlockChain), as well as Digital Signatures, and Zero Knowledge proofs. This chapter also clearly defines the overlapping regions and relationships between these study areas. The chapter ends with the identification of what makes this research the first of its kind.

Chapter 3 distills Chapter 2 into the main problems facing even state of the art identity management systems today, clearly identifies what they are lacking and the various entities that feel this the most. The chapter ends with how a solution will be used.

Chapter 4 details the scientific methods involved in this research and how the results and solutions were created. This chapter also summarizes the details for each phase of the research.

Chapter 5 presents the results of the research. It provides a new and generalized way of conceptualizing identity management systems, using formal definitions and cryptographic proofs that build the knowledge foundation for the main results. It also presents a hybrid solution (BlockChain Gateway & General BlockChain Identity), and provides a framework for using these to achieve the best of both worlds (Centralized and Decentralized Identities).

Chapter 6 concludes the thesis, summarizes the results, and compares these with literature to determine where the literature agrees and sections where previous literature disagrees. The chapter ends with suggestions for continuing this research and future developments.

Chapter 2

Related literature and theoretical focus

Digital Identity Management is not a new discovery in the Information Technology world. Since Ancient Roman times, Passwords have been the ultimate keepers of diversity and security [ALC+98] [Sto]. Even today, they are used to prove one's worth for obtaining some privilege others do not possess, but strongly desire to obtain. However, over the years with the emergence of the internet, Identity Management has transformed through several phases, although maintaining the same goals of:

- Verifying that a person is who they claim to be (*authentication*) and, consequently,
- Providing them access to some associated resource(s) or service(s) (*authorization*).

These two goals, authentication and authorization, are the backbone of *access control* [MW] and work together to generate the required trust necessary to provide resource(s) or services(s) to assumed unknown persons.

2.1 Stakeholders in Digital Identity Management

There are 3 main stakeholders (entities) involved in Digital identity management [SL]. Figure 2.1 shows these stakeholders and their respective roles.

Subject – An entity (individual or organization) whom the data is about, and/or actually owns the data and as a result would suffer losses if their privacy is violated. A privacy violation occurs when Subject's data is [BBHRSG17] [HB08]:

- Used by a third party that is not desired by Subject.
- Used in a period that is not desired by Subject and/or
- Used for a purpose that is not desired by Subject.

In short, Subject's main goal is to get some desired resource(s) or service(s) while simultaneously preventing a privacy violation. It should be noted that sometimes Subject may choose to relax their *privacy policy* to some degree, if they perceive that the gain (resource or service) is more valuable than some specific personal information.

Authentication agent – An entity that verifies that the Subject is who they claim to be or has what they claim to have. This could be a verification of whether Subject is registered (has subscribed) to a particular service, exceeded a certain age, has a particular nationality, is human and not a robot, etc. Typically (not strictly), an Authentication agent

- Stores the Subject's data pre-verification and also (issue claims)
- Provides a mechanism for Subject to verify their claims).

An example is a simple website login interface with a backend database. It should be noted that Authentication agent is usually an implicitly *trusted agent* (*someone Subject knows can violate their privacy policy without getting caught*) [BDP04]. The goal of this entity is to maintain trusted information about Subject and increase the number of Subjects they have if it increases its gains. Thus, the more data they have on Subjects, the more support they can provide for Authorization agents, hence, the more valuable they become.

Authentication can be done in the real world (fingerprint verification, DNA test, facial recognition) or virtually. When Authentication is done in the real world, it is usually referred to as identification. Identification can be expensive and if not done right can expose sensitive information, therefore, there is a need to minimize real world authentications as much as possible. Thus, performing them only when absolutely necessary.

Authorization agent – An entity that provides some resource(s) or a service(s) to a Subject after they have been authenticated. The main goal of this agent is to link the right person to their respective allowed resource(s) or service(s). This entity may obtain some profit from the Subject in exchange for the resource(s) or service(s) provided.

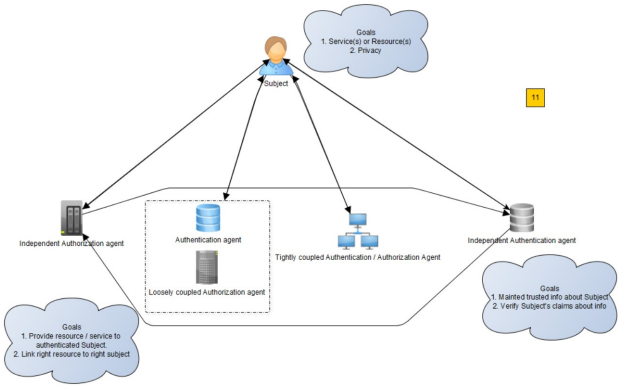

Figure 2.1: Stakeholders of identity management and their goals

2.1.1 Ideal world vs Real world

In an ideal world, all 3 stakeholders perform their duties accordingly without ever performing any action that can result in the suffering of another stakeholder. However, this situation is only achievable if all 3 stakeholders have the same goals or at least perfectly aligned incentives.

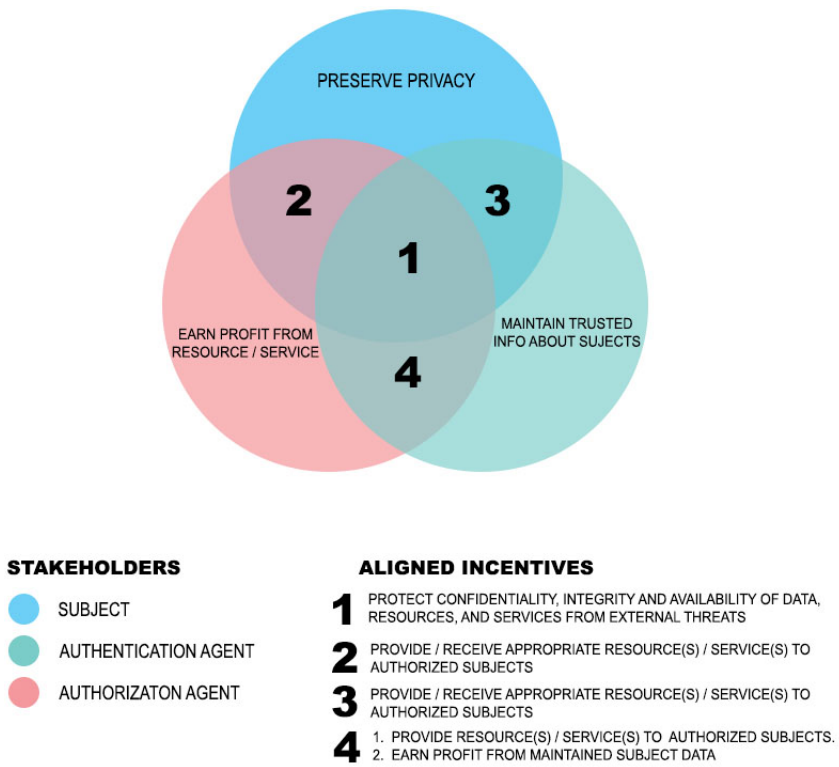

Figure 2.2: Incentive alignment diagram

Nonetheless, this is rarely ever the case in real life. In the real world, as can be seen from

Figure 2.2, there is a clear *incentive misalignment* amongst the three parties, with some misalignments being more severe than others.

The misalignment is most severe between Subject and Authentication agent, simply because they have directly opposing goals. Subject wants to preserve their privacy while the Authentication agent wants to benefit off their personal data. More so, this misalignment is even harder to reconcile because storing data does not necessarily mean you own it and owning data does not necessarily mean you always store it.

On the other hand, there is a strong incentive alignment (weakest misalignment) between Authentication and Authorization agents. These agents have more than one major aligned incentives as seen from Figure 2.2. In fact, their incentives are so aligned, it creates a mutual dependency (symbiotic relationship) between them. This makes them more likely partners.

In general terms, the stronger the incentive alignment, the greater the probability of collusion or partnership. In the same vein, the stronger the incentive misalignment (or the weaker the incentive alignment), the lesser the chances of collusion or partnership. Table 2.1 shows the probability of collusion between stakeholders expressed as a heat map.

STAKEHOLDERS	**Subject**	**Authentication agent**	**Authorization agent**
Subject	–	**Weaker Alignment**	**Weak Alignment**
Authentication agent	**Weaker Alignment**	–	**Strong Alignment**
Authorization agent	**Weak Alignment**	**Strong Alignment**	–

Table 2.1: Heat map showing probability of collusion between Stakeholders

2.1.2 Collusion amongst Stakeholders

As shown in Figure 2.2. all the stakeholders have their individual incentives for performing their tasks. It should be noted that in most real world applications, the Authentication and Authorization agents are not distinct and it is easy to see why from Figure 2.1. They may be tightly or loosely coupled.

In situations where the Authentication and Authorization agents are tightly coupled, there is a clear conflict of interest simply because, the agent providing services for Subject (Authorization agent) is the same agent storing the data (Authentication agent). This suddenly provides a perverse incentive to the Authentication and Authorization agent. There are

many of such *"double agents"* today such as Facebook, Google and many more.

Consequently, this creates a situation where such agents (acting as both Authentication and Authorization entities) use Subject's data for other purposes unintended by Subject such as selling the data (whether personal data or otherwise) to third parties. A practical example is the Facebook – Cambridge Analytica case [9]. In the *"double agent's"* mind, they are already benefiting from providing resource(s) or service(s) to the Subject. What is wrong with making a little more if the Subject never has to find out? Or if the subject doesn't really consider that data as sensitive?

On the other hand, in some situations, the Authentication and Authorization agents are loosely coupled, for example, logging into a Dating site using Facebook. Even in such situations, although the two agents have different incentives, those incentives are still aligned creating a mutual dependency. Hence, these agents are more likely to collude since it is much easier for them to form mutually beneficial partnerships. For example, in this case, Facebook provides the Dating site with more users and the Dating site provides more information about the activities and preferences of these users.

It is much harder for Authentication and / or Authorization agent(s) to collude with a Subject, owing to the fact that, although a Subject is the custodian of the data, this entity has the least leverage. As an individual Subject, data provided to Authentication and / or Authorization agents is negligible in value compared to the resources received. Therefore, and individual Subject has the least leverage and hence less chances of collusion with the other two agents. However, as a large group of Subjects, there may be enough leverage to form a collusion.

In summary, collusion form based on leverage and incentives. As a general rule of thumb, there is an inversely proportional relationship between the number of roles of Authentication and Subject's control over their data. Thus, the more roles Authentication agent has, the less control Subject has over their data and the larger the chances for collusion between Authentication and Authorization agents.

2.1.3 General Problems in Digital Identity Management

There two broad categories of problems that may result in Digital Identity Management Systems depending on the system's configuration:

Efficiency and Safety Problems:
It is sometimes possible (actually preferable) for a single physical / mundane entity (person or organization) to have different kinds of data (*virtual identities*) across different organizations (Authenticating agents). For example, Bob can have his information in a school database and also an insurance database. This entity-identity relationship is shown in Figure 2.3.

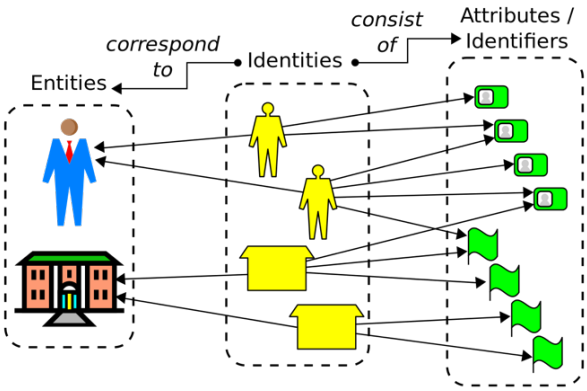

Figure 2.3: Physical vs Virtual Identities

If these two Authenticating agents (School and insurance agent) do not have the same database schema, which is usually the case, it can be rather difficult for them to share information about Bob. This is especially true when there are irregularities (*Example: Similar names in database or Bob registers with a different email address or phone number in each organization because he doesn't want his identity to be tracked between organizations*) and / or when there are many Authenticating agents. In such edge cases, the problem becomes exponentially harder, when one entity (such as governments) is required to maintain and track all data and activities of entities.

Physical (mundane) identities and virtual identities have a many to many mapping, thus one mundane entity can possess many different virtual identities and many mundane entities may possess one virtual identity. [Bir] This mapping is one of the major root causes of digital identity management problems.

Data sharing is, therefore, one of the major efficiency problems of digital identity management, because even at the atomic level, it requires that the two parties (Authenticating agents) sharing the data resolve the differences in their database schema. Unfortunately these two parties do not always have the luxury of being able to synchronize database schemas as this usually requires extra information which is not present in either databases.

Governments and large organizations can be likened to a super Authentication agent that manages a giant cluster of Authentication agents. These entities suffer the most from the data sharing problem because not only are their multiple Authentication agents supposed to share information amongst one another, they (the Government) are also required to keep track of which data (virtual identity) belongs to which individual/organization (physical identity) and what activities they are performing at some given time.

If the multiple virtual identities are efficiently reconciled with the corresponding physical identities, then governments and large organizations succeed in their job of maintaining trusted information about individuals, organizations, assets and activities. If not this will

result in slow synchronization of changes and a high error rate.

In summary, the efficiency problems above create a safety issue, where a single mistake could lead to errors in different databases or potential loss of data. The setup of this system (multiple data silos), also opens up multiple avenues of possible attack (broad attack surface).

Security and Privacy Problems:
Arguably, safety is a loose form of security requiring weaker constraints. However, a safety issue can easily be turned into a security problem because a rational adversary (*an adversary aiming to achieve the best results with least resources*), can choose specific safety issues against a system, thus making them a bigger security issue.

Generally, whenever an incentive misalignment occurs (*with or without collusion*), there exists a possibility (or *threat*) of such an action (*attack*) that is beneficial to the initiating party (*called attacker(s)*) and harmful to some other party (called victim(s)). A threat that poses the least exploitation challenge to an attacker (path of least resistance) is called a vulnerability (or *Dominant Attack in the victim/defender's point of view*).

Depending on the nature of the threat, any of the stakeholders could be victim(s) or attacker(s). It should be noted that if there is an external attacker, then it is possible for all 3 stakeholders to be victims. Based on the nature of the threat that arises in the Identity Management System, we introduce 3 configurations of attacks (not mutually exclusive) that may occur:

- **Type 1 Attack**
  In this kind of attack, all 3 stakeholders (Subject, Authentication, and Authorization agents) are victims and there is an external adversary / attacker. If there is such a situation where an external attacker gains unauthorized access to the database of the Authentication agent, this would be a classic example of a Type 1 attack.

  The Authentication agent loses their credibility because the database can no longer be trusted, since its integrity may be compromised. Subjects immediately lose their privacy since the confidentiality of their private data is breached and the data may be use for any purpose and at any time the attacker chooses. Lastly, the Authorization agent may be tricked into providing resources or services for unintended unauthorized parties. All 3 stakeholders become victims from a single attack.

  In summary, any configuration that allows a central store of data increases both the value and risk of that data store. In such a configuration, the Authentication agent will also service as the single point of failure since an attacker can do the most damage from a single attack.

- **Type 2 Attack**
  In this kind of attack, the Subject is the victim and the Authentication and / or Authorization agent(s) is / are the attacker(s). If there is such a situation where either the Authentication agent or Authorization agent or both collude to use the Subject's data in a manner which violates the Subject's privacy, this would constitute a Type 2 Attack.

  Type 2 attack does not only involve attacks on privacy. For example a subject may register to a service provider with their email address and password, and if it is possible for the Authentication agent to use this information to access another resource of the Subject (example their email), then this constitutes a type 2 attack.

  In summary, a type 2 attack occurs when any information provided by Subject is used in such a way that ends in negative consequences for Subject.

- **Type 3 Attack**
  In this kind of attack, Subject is the attacker and Authentication, another Subject and / or Authorization agent(s) is / are the victim(s). If there is any situation where a Subject can compromise the integrity / credibility of Authentication and / or get access to unauthorized resources, this would constitute a Type 3 attack.

  There are several forms of type 3 attacks. Some examples are as follows:

  1. An individual creating more than one virtual identity in such a way that makes Authentication agents perceive the created virtual identities as multiple physical identities. This is commonly known as Sybil attack named after a book written by Flora Rheta Schreiber in 1976 describing a woman with multiple identity dissociative disorder. A Sybil attack is generally geared towards compromising the reputation policy of the Authorization agent's service. Example, if it were extremely easy to create YouTube profiles, one could easily perform a Sybil attack creating millions of profiles and having them subscribe to the same channel, creating views or likes. This increases the reputation of the Subject and diminishes the reputation of the service.
  2. Identity theft is also another form of Type 3 attack. This is caused by compromising the Integrity of Authenticating agent in manner in which Subject can assume the identity of another Subject, thus causing harm to to another Subject (victim). In this case the Authenticating agents and / or Authorization agents become collateral damage.
  3. Fake (Ghost) identities are also a notorious form of Type 3 attacks, where a Subject creates a false identity usually to trick other Subjects. This is also a major attack on other Subjects resulting in the compromise of the credibility of Authenticating agent (collateral damage).

In general, Type 3 attacks are the hardest to defend against since in many identity Management Systems, it is hard to determine which physical entity matches what virtual identity / identities (*Identity Mapping Problem*).

It should be noted that there are other configurations of attacks such as Authentication agent as attacker and Authorization agent as victim or vice versa, however these attack types are less prevalent as at the time of writing as compared to the 3 discussed above.

2.2 Evolution of Digital Identity Management

2.2.1 What is a digital Identity?

There are several situations today in which an entity (person or organization) in possession of some personal information wishes / needs to prove the credibility of that information to a third party (stranger). Some examples are as follows:

- Proving whether you are a real person or not (robot, AI) over the Internet
- Proving at the airport control zones that you are a citizen of a country
- Proving in an exam hall that you are a member of an institution
- Proving that you are over 18 years

In some situations where there is some degree of trust between the prover and the verifier, it is easier to come to an agreement. Conversely, in situations where strong authentication is needed or where there is zero trust prior to authentication, it becomes exponentially difficult to come to an agreement especially when verification is conducted over the Internet.

As clearly seen, it is not always easy to prove that you are who you claim to be or that you actually have something you claim to have. This is a major prerequisite of Identity management systems; to generate trust between a prover and a verifier when there is reason to doubt the credibility of some claim(s).

A set of attributes (e.g name, date of birth, etc.) or information related to an entity that is used by computer systems to represent an external agent is known as a digital identity [ALC+98]. To make it clearer, this set of attributes is related to a Subject, and this information is used by Authorization or Authentication agent(s) for their respective purposes as described in Section 2.1 In essence, a digital identity is just a claim that is verified by an Authentication agent.

In the traditional case, a verified claim (digital identity) is only used by a single Authorization agent related to the verifying Authentication agent, however, sometimes it is beneficial if a single digital identity (claim verified by a particular Authentication agent) is

reusable with several Authorization agents globally. This creates a seamless user experience and opens flexibility in adding new services or resources.

What is important to note here is that there is no need for the Subject to be individually trusted by all of the authorization agents. They can provide their resources / services based on the sheer fact that one Authenticating agent trusts Subject. **In summary, trust is transitive**. If entity **A** trusts entity **B**, and entity **B** trusts entity **C**, then entity **A** in essence can trust entity **C** to **some degree**. It should be noted that this is not unintentional trust or reliance as stated by [CH96], but rather a formal definition has been given in section 5.1.4

Most digital identity issues occur because of poor reconciliation and synchronization of information (verified claims) when trust needs to be shared. This reconciliation typically involves propagating changes, reconciling differences, and mapping physical identities to their matching virtual identities. To understand the extent of this problem, we propose a puzzle / scenario shown below.

2.2.2 The many Clones problem

Figure 2.4 illustrates the many clones problem. In a world where creating human clones is possible, imagine you had many clones of yourself in many different countries. Each clone has different interactions and experiences.

- What is the most efficient way for your original self to keep track of the knowledge and experiences of all clones?
- What is the most efficient way to update all / some clones with some particular information when necessary?

Figure 2.4: The many clones problem

There are many possible solutions to this puzzle. These are some of the possible solutions:

1. Central Information store – One typical solution is for each clone to regularly report knowledge to a central information store. The original person can keep track of all knowledge and experiences from the central store. Clones may also update their information from this central store when it becomes necessary.

2. Multiple Information stores – A selected group of clones with similar experiences or knowledge regularly update an information store dedicated to them. The original person can keep track of all knowledge experiences. Updates may also be made to the appropriate information store.

3. Direct Sharing With Master Control – Each clone maintains a private information store of their knowledge and experiences. If the original self needs to keep track of a particular clone, a direct communication (Say phone call) is made with the clone and original self is updated. If there is a need for some clones to update one another or original self to update any group of clones, the necessary number of communications. However, all activities are initiated by the creator of the clones (original self).

4. Linked Clones – Another solution is to somehow link all the clones. Each clone maintains a private information store of their knowledge. If the original self needs to keep track of the clones, making contact / communication with one clone is enough. Since all of them are linked, the information is propagated throughout the chain of clones. Updates are performed the same way since all clones are linked. The critical difference here is that, since every clone is linked, all clones have full control / autonomy and may arbitrarily choose when to update or communicate.

Table 2.2 below shows the advantages and disadvantages of each solution.

Solution / Analysis	**PROS**	**CONS**
Centralized Information Store	1. Simple and fast updates. 2. Easy synchronization	1. Single point of failure. If Information store is down, all knowledge about clones are lost. 2. Common point of trust needed. Entity acting as data store must be trusted by all clones and original self. 3. Inflexible design as clones may store different kinds of information and experiences 4. Slow search and large database. 5. Redundancy of duplicate entries
Multiple Information Stores	1. Simple and fast updates. 2. More flexible 3. No single point of failure 4. Faster search multiple databases.	1. Large attack surface. Each Information store is another way to get potentially sensitive information. 2. No common point of trust. 3. Synchronization between information stores may be difficult 4. Redundancy of duplicate entries

Direct Sharing With Master Control	1. Less storage required because each clone maintains their own database. 2. No common point of trust needed. 3. Flexible design 4. No central point of failure. If one clone is down, the other clones are still active.	1. For the whole system to be updated, almost n2 communications are needed. Each clone updates (n – 1) peers. Thus n * (n – 1) necessary communications. This is a lot of bandwidth. 2. It is slow and prone to errors as some updates may fail. 3. Clones have no autonomy or control.
Linked Clones	1. Simple and fast updates. 2. Easy Synchronization. 3. No single point of failure because of information propagation 4. Fast search. 5. Flexible 6. Less bandwidth 7. Smaller attack surface since attack one clone is equivalent to attacking all clones 8. Persistent 9. No common point of trust required. 10. Clones have full control or autonomy.	1. Information propagation may take time to happen. 2. Every clone participates even in the simplest operations since all of them are linked.

Table 2.2: Advantages and disadvantages of each solution

The *"many clones problem"* is a variation of the Byzantine General's Problem [LSP82]. In this paper, it has been put in this form to better understand the problems of data sharing amongst Authenticating and / or Authorization agents in Identity Management and identity mapping problem.

If the problem being analyzed is the data sharing problem, a clone in this context refers to the Authenticating Agent. Each clone being in a different location may be taken literally or that the Authentication agent exists on a different server.

On the other hand, if the problem being analyzed is the identity problem, then a clone in this context refers to group of related attributes (name, email, gender, etc.) about a physical entity (Subject) which forms its digital identity on a computer system. Each clone being in a different location represents the data being held by different Authenticating agents.

General identity management solutions usually solve the "many clones problem" in a particular way. This has led to four (4) main classes of Identity management Systems: Centralized, Federated, User-Centric, and Self Sovereign Identity.

2.2.3 Centralized Identity

At the genesis of the internet, organizations like IANA determined the validity of IP addresses in 1988, ICANN arbitrated domain names in 1998, and by 1995 certificate Authorities (CAs) stepped up to help Internet commerce sites to prove that they were who they claimed to be. [All]

One common feature shared by all these organizations is that they were centralized authorities [FP12]. They alone had the power to determine whose identity could be trusted or not. This means that they could also deny anyone's identity, or perform false verifications. Another problem noticed in that era was that as more and more websites were created, identities became balkanized, and users were forced to juggle multiple identities for different websites.

To alleviate this problem, some organizations took a small step beyond centralization by creating hierarchies. Due to the heritable transitive nature of trust, a root Authentication agent can authorize subordinate Authentication agents to oversee their own hierarchy. However, this was not a complete solution because root Authentication agents still hold the core power of the system, just that now the power is delegated in smaller amounts to centralizations beneath them. [All]

In summary a centralized Identity is characterized by administrative control by a single authority or a hierarchy [All]. This is similar to the centralized information store solution to the "many clones problem". To a large extent, identity on the Internet today is still cen-

tralized — or at best, hierarchical. Digital identities are owned by CAs, domain registrars, and individual sites, and then rented to users or revoked at any time.

2.2.4 Federated Identity

In 1999, Microsoft's Passport initiative was one of the first to de-balkanize digital (online) identity in a new way[All]. It made it possible for users to use a single identity on multiple websites. Such an identity is called a federated identity. However, Microsoft was at the center of the federation which essentially made it almost as centralized as the traditional centralized authorities.

To remedy this situation, Sun Microsoft organized the Liberty Alliance (2001) to resist the idea of centralized authorities. [All] They intend to create a true federation, however, the result was oligarchy: The power of centralized authorities was now divided amongst several powerful Authentication agents. This improved the problem of balkanization, however, each individual website remained an authority.[FP12]

In summary, a federated identity is characterized by administrative control by multiple, federated authorities.[All] This is quite similar to the multiple information stores solution to the "many clones problem".

2.2.5 User-Centric Identity

In 2000, the Augmented Social Network (ASN) designed the foundation of a new form of identity for the next generation of internet. In their paper [JHF03], they proposed building a "persistent online identity" into the very fabric of the internet. ASN addressed the issue of Passport and Liberty Alliance, arguing that they could not meet the goals of a persistent online identity because the "business-based initiatives" put too much emphasis on the privatization of information and the modeling of users as consumers.

After this foundation established by ASN, the Identity commons, in 2001, began to consolidate the revolution of digital identity with a focus on decentralization. They teamed up with Internet Identity Workshop (IIW) in 2005, in a series of semi-yearly meetings to advance decentralized identity.

Instead of the server-centered model of centralized identities, the IIW proposed a new term, called user-centric identity. User-centric model basically puts the user at the heart of the identity process, thus creating a better user experience. Soon this meaning was expanded to include providing the user with control over their identity and decentralized trust.

Many digital identities were created through the work of the IIW, including OpenID in 2005, OpenID 2.0 in 2006, OAuth in 2010, FIDO in 2013, and OpenID Connect in 2014

[SPM+11]. As a general rule of thumb, these user-centric identities focused on user consent and interoperability. This allows a user to decide which service they want and share their identity from service to another, thus debalkanizing their digital self.

The goal of user-centric identity grew in hopes of providing users complete control over their digital identities, however, these efforts were thwarted by powerful institutions. It is theoretically possible for users to have full control over their identities. For example with OpenID, a user could register its own OpenID, which he can then use autonomously. However, practically, there is a high entry barrier because created your own OpenID requires some technical savviness that casual internet users do not possess. Therefore, much like the federated identities, users prefer to use the OpenID from well known public websites, thus the final ownership of this identity will remain with the organizations that register them.

Apart from the IIW, Facebook Connect appeared in 2008 hoping to leverage the lessons learned from OpenID. However, it allowed only choice of provider which was Facebook, meaning that a user could connect to other websites with ther Facebook Identity. This meant even more user risk with using Facebook Connect as compared to OpenID, because if Facebook for whatever reason decides to close a user account, that user also loses all identities connected with that account. The methodology seemed user-centric, but by reduction, it is really centralized Identity (where the central authority is Facebook) all over again.

In summary, a true user-centric identity is characterized by administrative control across multiple authorities without requiring federation [All]. However, as we have seen, even with the best user-centric identities more needs to be done in order to give the user full control over their identity. This is similar to 'Direct sharing with Master control' solution of the "many Clones problem", except that each clone is controlled by an administrative authority and no clone has full control over their decisions.

2.2.6 Self Sovereign Identity

Even before the full development of the idea of self-sovereign identity, there were several hints. In 1991, Pretty Good Privacy (PGP) [Gar95] offered the first hint of what could become self-sovereign identity. It introduced the 'Web of Trust' [AR97], which establishes trust in a peer to peer fashion with each user being an introducer or validator of a public key. This solves the political problem of who gets to validate a public key, since everyone has equal chance of validating a public key (little or no entry barrier). 'Web of Trust' was a powerful decentralized trust management, however, it focused on email addresses, hence, it still depended on centralized hierarchies. For many reason PGP was not widely adopted.

In 1996, Carl Ellison wrote a paper titled "Establishing Identity without Certification Authority" [E+96]. He considered PGP and centralized authorities as possible methods for

defining digital identity. He settled on a protocol of repeated public key transmission and verifying the resulting identity with shared secrets over a secure channel. This allowed two parties to control their own identity without depending on authorities. In 2000, ASN and the liberty alliance also a hint of self-sovereign identity because their developments were based on the assumption that every individual ought to have the right to control their own online identity, and for the last two decades there has also been a growing push to return identities to the people, so that they could fully control them. [All]

In general, user-centric designs turned centralized identities into interoperable federated identities with centralized control, while also respecting some level of user consent about how to share an identity. Self-sovereign identity builds on this, thus, it provides interoperable decentralized identity, however, it also provides user autonomy. It makes users the rulers of their own identity and ensures that users have the final say when it comes to their identity.

The term Self-sovereign was first coined in February 2012, when developer Moxie Marlinspike wrote about "Sovereign Source Authority" [RS13]. This catalyzed the work of Patrick Deegan in March that year on Open Mustard Seed, an open-source framework that gives users control of their identity. Several initiatives then started geared towards advocating for individual rights to identity.

Since then Self-sovereign identity transformed from an idea, to mathematical policy, to legal policy, and now international policy especially in Europe due to the refugee crisis [All]. As the next step beyond user-centric identity, this means that apart from the user being central to the administration of identity, interoperability of a user's identity across multiple location (transportable), with user consent and true user control is required for user autonomy.

A Self-sovereign identity must allow ordinary users to make claims, which could include personally identifying information, group membership, or personal capability. This information may be asserted by other persons or groups. In achieving all this, it is essential to carefully protect individuals, defend against financial and privacy losses, and also prevent human right abuses. [Pre]

In conclusion Self-Sovereign identity is characterized by individual control across any number of authorities, providing a lifetime portable digital identity for any person, organization, or thing [All]. This identity does not depend on any centralized authority and can never be taken away, because the user fully controls it. It is similar to the "Linked clones" solution to the "Many clones problem".

2.2.7 Principles of Self-Sovereign Identity

Recent researches have made approaches towards providing self sovereign identity, these include, ID2020 by Microsoft [id2], Secure Key and IBM [ibm], Identity Chain [DT], ERC #752 [Vogb] [Voga], Sovrin [KRGP18], Estonian Identity [Ped13] , and many more. Most of these identities, with the exception of the Estonian identity use the blockchain.

However, it should be noted that there is a lot more to self-sovereign identity than just decentralization. In order for an identity to be considered as self-sovereign, it must meet ten (10) principles. This proposal and details for these principles are found here [All] and are summarized as follows:

1. Users must exist independently of the identity.
2. Users must be able to control their identities, thus be the ultimate authority on their own identity.
3. Users must have access to their own data within their identity.
4. All systems and algorithms regarding the identity must be transparent.
5. Identities must be persistent, preferably last a lifetime, or for as long as the user wishes.
6. Identities must be easily transportable, thus must not be held by a single third-party (entity).
7. Identities must be as widely usable as possible (interoperability).
8. Users must be given the chance to consent to the use of their identity.
9. Disclosure of claims must be minimized (minimization), thus, when identity data is used, only the minimum amount necessary to achieve the requisite task must be exposed.
10. In cases of conflict, the rights of the user must be protected.

In the light of the above, it is crystal clear the the evolution of identities right from centralized identities have gravitated towards self-sovereign (more decentralized) identities. Self-sovereign identities are the state of the art today in Digital Identity Management systems.

2.3 The BlockChain

Identity Management in most cases is achieved with cryptography. However, cryptography as a tool in general has its strengths and weaknesses. Thus there are somethings cryptography can do and other things no form of cryptography can achieve.

Cryptography can:

- Prove that a particular message originated from a particular entity (Message Authenticity). Digital Signatures and Certificates are used to prove this property.
- Prove that a particular message is linked to another message or that one message came before the other. Hashes and / or Merkle Trees can be used to prove this property.
- Ensure that a message is intelligible only to desired Entities (Confidentiality). Encryption and Decryption can be used to ensure this property.
- Ensure verify that a message is received or retrieved in its original form (Integrity). Message authentication Codes (MAC or HMAC) can be used to ensure this property.

On the other hand, Cryptography can **NOT**:

- Ensure that a message is available whenever requested (Availability). To ensure availability, you need another technique to ensure message persistence. An example is to use a distributed system (database).
- Ensure that some security properties (example Authenticity, Confidentiality, Integrity, and Availability) hold for a long time into the future. To ensure the persistence of security properties over a long period, one can use economic incentives.

To ensure persistence of messages (Availability) and persistence of security properties, it is essential to use a mixed solution of both cryptography, economic incentives, and distributed system for persistence. A solution that achieves such a hybrid form of persistence of messages (Availability) and persistence of security properties using economic incentives is called a Distributed Ledger Technology [Wal16]. The branch of Information Technology that combines cryptography with economic incentives is called Crypto Economics [Rab17].

A blockchain, therefore, is a special (persistent, transparent, append-only, and public) Distributed Ledger Technology backed by a consensus algorithm. The consensus algorithm is the economic incentive that ensures persistence of security properties through time.

2.3.1 Timeline of BlockChain

The concept of a distributed ledger in businesses like currencies, and property registries has been around for decades. In the 1980s and 1990s, a cryptographic technique proposed by David Chaum known as Chaumian binding, was used to provide a currency with a large degree of privacy. These anonymous protocols were called e-cash, but they failed to gain traction due to their reliance on a centralized intermediary [B$^+$14].

The era of economic incentives in decentralized systems begun in 1998, when Wei Dai's

b-money [Gru13] became the first proposal to introduce the idea of creating money and decentralized consensus through solving computational puzzles (cryptocurrency). However, this proposal could not be realized because it was scant of implementation details. [B+14]

By 2005, Hal Finney built on the ideas of b-money and Adam Back's computationally difficult ***Hashcash*** [B+02] puzzles to introduce a concept of reusable proofs of work (POW), which could serve as the foundation of a cryptocurrency. However, this proposal also fell short of the ideal because it also relied on trusted computing as a backend.

Finally, in November 2008, a paper was posted on the internet under the name Satoshi Nakamoto titled Bitcoin: A Peer-to-Peer Electronic Cash System [Nak08]. This paper detailed methods of using a peer-to-peer network to generate what was described as "a system for electronic transactions without relying on trust".

In 2009, a decentralized currency was for the first time implemented in practice by Satoshi Nakamoto, combining established primitives for managing ownership through public key cryptography with a consensus algorithm for keeping track of who owns coins, known as "proof of work". The mechanism behind proof of work was a breakthrough in the space because it simultaneously solved two problems.

First, it provided a simple and moderately effective consensus algorithm, allowing nodes in the network to collectively agree on a set of canonical updates to the state of the Bitcoin ledger. Second, it provided a mechanism for allowing free entry into the consensus process, solving the political problem of deciding who gets to influence the consensus, while simultaneously preventing Sybil attacks. It does this by substituting a formal barrier to participation, such as the requirement to be registered as a unique entity on a particular list, with an economic barrier - the weight of a single node in the consensus voting process is directly proportional to the computing power that the node brings.

Thus, by January 2009, the bitcoin network came into existence with the release of the first open source bitcoin client and the issuance of the first bitcoins, with Satoshi Nakamoto mining the first block of bitcoins ever (known as the "genesis block"), which had a reward of 50 bitcoins. The value of the first bitcoin transactions were negotiated by individuals on the bitcointalk forums with one notable transaction of 10,000 BTC used to indirectly purchase two pizzas delivered by Papa John's.

Since then, an alternative approach has been proposed called proof of stake (POS), calculating the weight of a node as being proportional to its currency holdings and not computational resources. The relative merits and demerits o; the discussion of the relative merits of the two approaches (POW and POS) are discussed in the next chapter.

2.3.2 State and History

As a conceptual aid, it is sometimes necessary to visualize the BlockChain as a state transitioning system [Dam] [B+14]. A state is like a balance sheet. It shows what we should be concerned about at any point in time.

In Bitcoin, the state is just the amount of money everyone has at any given moment. Technically, these are the Unspent Transaction Outputs (UTXO). In Ethereum, a state is strictly not made up of transactions. In fact transactions constitute history. Instead, every user has an account that tracks their state.

There are two types of Ethereum accounts: User accounts (controlled by mundane users) and Contract accounts (controlled by code). Therefore, an Ethereum state is made of various account objects. An account object is made up of the account balance, account nonce, account storage, and contract code (if any).

2.3.3 Layered View of the BlockChain

To better understand what the BlockChain is and the role it can play in Self-Sovereign Identities, it is essential to understand the basic building blocks of any BlockChain. To this effect, we have provide Figure 2.5. which shows a layered view of the BlockChain.

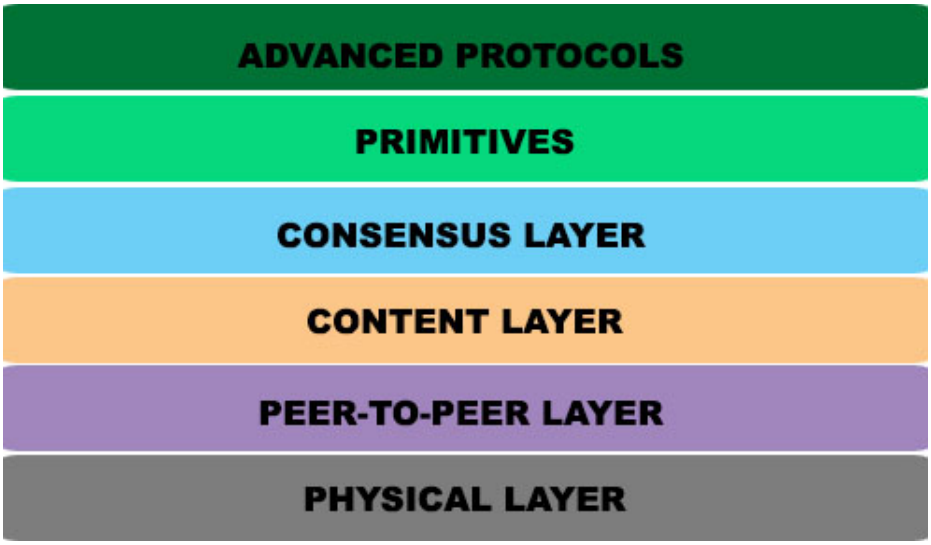

Figure 2.5: Blockchain

Physical Layer
This layer is quite similar to the physical layer of the OSI [BP04]. However, in the blockchain perspective, it is made up of disorganized computers. These devices are not limited to Personal computers.

Peer to Peer (P2P) Layer
This is the first level of organization in a blockchain. At this layer a protocol for a peer to peer network is established for communication between disorganized computers [DW13]. This protocol achieves objectives such as adding new peers and assigning GUIDs, removing peers, routing data through the network, routing table management, and many more. There are several algorithms for peer to peer networks. They include Chord, Pastry, etc.

If a BlockChain is open for any computer can join in at this P2P layer and perform any actions (transactions & mining), this is called a permissionless BlockChain. Two typical examples are Bitcoin and Ethereum. However, in some cases, it is desired that some special privileges be reserved for special nodes, and not everyone can join in the physical layer of the block. This kind of BlockChain is called a permissioned BlockChain. A typical example is HyperLedger.

Content Layer
This layer is concerned with the data structures that hold BlockChain data on each node in the peer to peer network. In Bitcoin and Ethereum data is stored in blocks with each block pointing to the previous block. The first block points nowhere and is called the genesis block. Typically this block is hard coded. This data structure is quite similar to a linked list. In other blockchains like IOTA [DB18], a Directed Acyclic Graph(DAG) [SdSSW95] is used which is commonly known as a Tangle. Whatever data structure at the content layer acts as the ledger that is distributed among nodes in the BlockChain.

The content layer may also denote the type of storage (usually Distributed Hash Tables (DHT)) [KK03] used by the BlockChain. For example SWARM [DMH17] architecture used by Ethereum.

Consensus Layer
The consensus layer ensures that every peer in the network agrees on the same content at any given time using algorithms called consensus protocols. Some of these protocols include Proof of Work, Proof of Stake, etc. [ZXDW16] More about consensus is discussed in the next section.

Primitives (State, transactions, and state Transition function)
A BlockChain can be viewed as a State transitioning system. In a BlockChain there exists a conceptual state transition function F, such that $F(S, T[n]) \rightarrow S^I$, where S is a valid old state and S^I is a valid new state. T is an array of n transactions to be added to the new block.[Woo14] The state transitioning functioning enforces the properties of the consensus layer. In Bitcoin, F ensures that each transaction in T is unspent, has a signature matching the public address of the UTXO, and that transactions do not create value, thus, the sum of all input UTXO is greater or equal to the sum of all output UTXO in the transaction.

Only coinbase transactions create value in the form of miner rewards. Also, if input UTXO is greater than output UTXO, the difference is considered the transaction fee.

Lastly, each transaction has a unique ID which is usually a cryptographic hash [4] of the content of the transaction. This unique ID can be used to verify the validity of any transaction. However, not every node has the luxury of maintaining a database of all transactions (and state). Such a node is called a full node. A majority of nodes, known light nodes, instead of maintaining a database of all transactions (and state) use a technique called a Merkle tree, so named after its creator Ralf Merkle. to validate transactions.

Merkle trees perform consecutive hashes of transactions (which are the leaves of the tree), iteratively pairing and rehashing, until one hash is formed. This is known as the root hash. Therefore, for a light node to verify a transaction, all they need is the transaction ID and the Merkle Proof of the transaction. This information can be obtained from a full node.

Merkle trees are extremely efficient and secure depending on the Hash function used. In Bitcoin a double SHA256 hash function is used mainly to provide a higher collision resistance [XLD05]. After the successful attack (collisions found at 2$\hat{6}$5 instead of expected 2 $\hat{8}$0), there was a growing fear among the Bitcoin designers that SHA256 had inherited the same weaknesses since the design of SHA256 is also based on the same Merkle-Damgard construction as SHA1. Therefore two rounds of SHA256 is expected to provide a higher collision resistance.

Ethereum, however, uses a different hash function altogether called Keccak256, which is based on sponge functions[BDPVA09]. It should be noted that Keccak256 is strictly different from SHA3[Cona].

Advanced Protocols
At this layer the primitives are used in innovative ways to create value in use cases. A few of these use cases are:

- Smart contracts – accounts that are controlled by program code (autonomous agent) such that each transaction to the Smart contract executes the code in the contract on all nodes [Woo14]. Input data may be provided in the transactions to Smart contracts.
- Atomic swap – Exchanging currency between two blockchains [But16]
- Payment Channels – Creating an offchain way to make transactions and executing settlements on the BlockChain.

2.3.4 Consensus (POW vs POS)

In a decentralized system like the Blockchain, it is usually desirable for all participating nodes to share a common database. This is called a distributed / shared ledger. However, because every node has access to possibly varying pieces of information (different versions of the ledger), there is a possibility for involved parties to cheat. We can NOT assume that all participating parties are honest, therefore, if a Blockchain is to prevent cheating of any form, three things have to be clearly defined: Consensus, State, and History.

If the content of the shared ledger has economic value especially, when the content of the ledger represent actual money, the first order of business is to establish a fair and open protocol for all participating parties to agree on the content of the shared ledger. Such a protocol that allows trust to be generated amongst parties that share mutual distrust for one another is called a consensus protocol. It should be noted that a consensus protocol may simultaneously achieve the following objectives.

- Establish an agreement of at least 51% of the participating nodes before any new information is added to the BlockChain. This is essentially a voting process.
- Ensure that it is computationally impractical or improbable for any one node or less than 50% of the participating nodes to change the content of the ledger. Thus, in effect the ledger should be append only and permanent.
- Incentivize participating parties to **honestly** partake in voting process that adds new information to the distributed ledger. Thus, provide some sort of reward to sustain these properties over time.
- Control the growth rate of the Blockchain.

A simple rule of thumb is that a good consensus protocol should easily generate user trust and prevent cheating. This is because even though the participating users do not trust one another, they each hold the belief that the chosen consensus protocol is fair, transparent, and protects their best interests. This allows the participating users to work together without ever needing to trust one another.

At the time of writing, there are two broad categories of consensus protocols: Proof of Work (POW) and Proof of Stake (POS). Each of these protocols have their relative strengths and weaknesses.

In POW, consensus is achieved through a randomized challenge that no one person can solve consistently better than others, thus NO single node has an advantage over others. Practically this is achieved as follows [ZXDW16]:

1. Select transactions from the transaction pool, verify them and construct them into a block.

2. Calculate the merkle hash tree, note the root hash.
3. Construct the header of the block: header of the most recent block (verify first), root hash of transactions, timestamp, and nonce.
4. Hash the header
5. If the hash meets the current difficulty (starts with a given number of zeros), send the block to everyone. If not, select a new nonce and recalculate the hash until it does.

A hash function in theory is a one-way function that takes arbitrary input and returns a fixed length output. If the above POW protocol is followed, it is very easy for each node to verify the correctness of the provided Nonce and transaction list by performing a single hash. Once this is done, it is added to their version of the Blockchain. On the other hand, in order to find an appropriate Nonce, one has to perform an astronomically large number hashes. The processes of finding the right Nonce from the POW protocol is called Mining.

On matured Blockchains, a single node finding an appropriate Nonce is highly impractical and improbable. Therefore, since all untrusting nodes now share the same problem of finding Nonces, collusions begin to form simply because they stand to benefit from a large reward if a solution is found. Currently, there are two types of POW protocols. The Bitcoin POW and the Ethereum version of POW called EthHash. Table 2.3 shows the diferences between Bitcoin POW and EthHash.

Bitcoin POW	EthHash
Uses a double SHA256 hash function for finding appropriate Nonce.	Uses keccak-256 hash function for finding appropriate Nonce.
Can be parallelized, since it requires only computational power. Thus one can build Application Specific Integrated Circuits of FPGAs to reduce search time.	Can NOT be parallelized, because it is memory hard. Hashes also require a 16mb cache in order to find correct Nonce.
Prone to centralization as miner pools control large portions of the Blockchain	Comparatively not as prone because the same mining equipment used in Bitcoin POW will not work for EthHash.
Mining difficulty is adjusted every 2016 blocks.	Difficulty renewed every hard fork.

Table 2.3: Differences between Bitcoin POW and EthHash

POW protocols make blockchains very hard to defraud or alter after the fact. This is simply because changing the content of any individual transaction will require the attacker to recalculate the hash to meet the difficulty, and if there are any parent blocks, recalculate the hashes for those as well. This is an expensive task for any one person, and on mature blockchains particularly difficult for a group of nodes who do not form the majority. At the current EthHash target of $2\hat{1}87$, the network must make an average of $2\hat{6}9$ tries before a

valid block is found.

POS protocols are very different from POW protocols. POW protocols in general require a lot of electricity for the computation. In fact, according to Digiconomist [Diga], Bitcoin mining alone consumes about 54 TWh of electricity per year. This is enough energy to power 5 million household in the US, or even power the entire country of New Zealand or Hungary. Apart from this POW, generally has a tendency to get centralized due to mining pools.

To solve these problems, a Bitcoin forum user known as Quantum mechanic proposed POS in 2011 [KN12]. The basic idea being Proof of Stake is an election process that one node to validate the next block. This node is called a validator, and instead of mining the process is called minting / forging. To become a validator a node has to deposit a certain amount of coins into the network as stake. This is the incentive that keeps validators honest, simply because if a validator approves fraudulent transactions, it can easily be detected and the faulting validator will lose a large portion of their stake. This is true because the possible gains from validating a wrong block is always less than the validator's stake (cost of cheating).

Even Proof of Stake has its own problems such as the fact that the election process is not open to everyone, thus, not everyone can become a validator. Secondly what if a node chosen as a validator refuses to do their work. Lastly both POW and POS have scalability issues, thus, the number of transactions that can be handled per second.

2.3.5 BlockChain Generations

Over time, 1st generation blockchains (e.g. Bitcoin), which are typically centered on a cryptocurrency became widely adopted, thus the value of Bitcoin grew exponentially [FH]. However, it quickly became clear that the technology backing cryptocurrencies (Distributed Ledger / Blockchain) was in itself very powerful , and could be used for other purposes.

This gave birth to 2nd generation blockchains (e.g. Ethereum) [FH]. Ethereum provided four main things that its predecessors could not provide[B^{+}14]:

- **A Turing complete programming Language**: Bitcoin has a weak version of a scripting language called Bitcoin Script. In fact basic public key ownership and *MultiSig* operations are performed by this Script. However, the main issue with any scripting language open to the public is the **Halting problem**. Thus given any program, it is almost impossible to predict prior to running, how long the program will take to finish execution or whether the program will finish execution at all. Thus the program may be an infinite loop or a disguised infinite loop based on some input supplied by the user environment. To solve the halting problem, Bitcoin completely removed loops from the programming language, hence, if you wanted to implement

an alternative elliptic curve signature algorithm, one would have to individually include the code for 256 rounds of multiplication. This was clearly impractical for use as a general purpose programming language. Ethereum found a new solution to the halting problem, thus, by metering the amount of computational resources (execution steps and storage) used, one can provide a deterministic prediction of the runtime of a program, and more importantly control it. This deterministic metering of computational resources is called ***gas***. As protocol economics dictates, one has to pay for gas if they want their transaction to be included in the public ledger. In the same light, if you are using for example an Ethereum based private network, it is not necessary to pay for gas. With the introduction of gas, Ethereum introduced a Turing complete programming language.

- **Implementation of arbitrary State**: Bitcoin does not allow the implementation of new arbitrary states. Ethereum on the other hand allows the creating of arbitrary states in Smart Contracts.

- **BlockChain Awareness**: UTXO in Bitcoin are blind to BlockChain data such as the nonce, the timestamp and previous block hash. This severely limits applications in gambling, and several other categories, by depriving the scripting language of a potentially valuable source of randomness.

- **Value Awarenes**: Sometimes there is the need for an organization to create a hedging contract, where for e.g. Bob and Alice deposit 5000 euros worth of BTC and after 30 days the blockchain script sends 5000 euros worth of BTC to Bob and the rest to Alice. This is notoriously difficult to implement with Bitcoin because UTXO are all-or-nothing. The only way to achieve this is through the very inefficient hack of having many UTXOs of varying denominations (eg. one UTXO of 2^k for every k up to 30) and having the contract pick which UTXO to send to A and which to B. However, this is very easy to implement with a simple Ethereum smart contract. To summarize, there is no way for a UTXO script to provide fine-grained control over the amount that can be withdrawn. This is called value blindness.

In September 2017, a new cryptocurrency Cardano came out claiming to be a 3rd generation blockchain [FH]. It is rather different from other cryptocurrency projects because it was built around peer reviewed papers. Instead of just writing a whitepaper and developing a solution, Cardano made sure experts from around the world read their papers and improved them before developing a solution.

The main distinguishing feature of 3rd generation blockchains is the massive improvement in Scalability, interoperability, and Sustainability. Bitcoin currently maxes out at a maximum of 7 transactions per second. Ethereum on the other hand maxes out at 15 transactions per second. VISA handles about 24,000 transactions per second. Apart from some major security issues, Cardano's Ouroboros and IOTA's tangle bring us one step closer to this transaction speed. More on scalability, interoperability, and sustainability can be

found here [FH]

Therefore, it is crystal clear that for a blockchain to become a global currency, it must meet these requirements, especially, if it is to be used as a global Identity Management System.

2.4 Why Identity Management Matches Blockchain

The BlockChain by design is decentralized, thus, it provides a centralized approach to Identity Management. In most state of the art systems, users store their identity in their personal decentralized wallets thus eliminating the problems of centralization. This decentralization also provides a large degree of mobility and user control to such identities.

A blockchain Identity also automatically provides a clear separation between the roles of Authentication agents and Authorization agents, hence degrading the probability of collusion of these agents against a Subject. In most cases Authentication and Authorization agents may be completely decoupled, thus removing the incentive to misappropriate Subject data.

Lastly, trust scales very well in the BlockChain environment as compared to traditional central server solutions. This is simply because, new entrants into the BlockChain Identity only need to trust the consensus algorithm as compared to trusting a server or even a group of servers.

2.5 GDPR vs Blockchain

On May 25th, 2018, the General Data Protection Regulation (GDPR) became enforceable in the EU region. This has a paradoxical effect on BlockChain data in general and should be strongly considered when implement any BlockChain Identity solution within the EU region.[**?**]

In the official documentation of the GDPR [Conb], the term 'erasure' and 'erase' are found 12 times. This goes contrary to the operation of the BlockChain since all data stored on the BlockChain are permanent. This concept is also reinforced in Art. 17 'Right to be Forgotten', however, 'erasure of data' is not clearly defined in the entire document. This provides the legal premise for the strict interpretation of the term 'erasure of data' or 'erase'. Consequently, even encrypting personal data and burning encryption keys is not acceptable as 'erasure of data'.

Another important aspect of GDPR on blockchain is the exfiltration of data from the EU. This is a major problem with public blockchains, since there is no control on who hosts a node. This is less an issue when it comes to *private* or *permissioned blockchains*.

Therefore, new protocols need to be established by BlockChain based Identity Management systems to compensate for losses in user privacy whether regulated or unregulated.

In summary, it is clear that enforcing data privacy and protecting user data is no longer optional in Europe. However, it should be noted that the text of the GDPR documentation is void of both technical and non-technical implementation details necessary to achieve GDPR compliance. This constitutes a significant research gap that this research aims to fill.

2.6 What makes this research new?

Recent researches have made approaches towards providing self sovereign identity, these include, ID2020 by Microsoft, Secure Key by IBM, Identity Chain, ERC #752, Sovrin, Estonian Identity, and many more. Apart from the Estonian Identity many of these use the blockchain, however, just using the BlockChain does not guarantee Self-Sovereign identity out of the box. However, these solutions do not completely meet the requirements of a self-sovereign identity. The Estonian Identity is initiated by the government, thus, all key pairs are created by the government. This reduces the transparency of the Identity, because apart from the fact that the government can generate ghost keys or misuse identities, purely accidental events could happen, such as accidently generating a large number of weak keys. [33].

Another self-sovereign identity is the Ethereum proposal ERC #752 and #753 which proposes a generalization Identity Management based on claims for Ethereum. Even though a description of how a claim is created and managed, little or no information is provided on how transparency and user control is achieved and maintained.

Once a user shares a claim with a third party (service provider) in the verification phase of a claim, they also lose their control and transparency after the fact. Thus, when a third party (service provider) is verifying a user's claim, the user should be able reduce the amount of information attributes provided in the verification process, how long the information is exposed, and be kept updated about any and all transactions involving their claims.

This research begins to build the foundation of how a fully self-sovereign identity can be achieved, including the achievement of transparency and control which are the cornerstones of user privacy.

Chapter 3

The problem

3.1 What is the need?

Recent technological developments have increased the use of more anonymizing technologies like Distributed Ledger Technologies for everyday business processes. This is largely because of the increased security, scalable trust, efficiency, zero infrastructure, and the other advantages a business enjoys when they start BlockChain based organizations (Distributed Autonomous Organization). On the user front, one reason for the increased adoption is the growing awareness of users about their privacy [five compelling cases coincatalys].

Privacy enhancing technologies like the BlockChain have seen a large influx of revenue over the past years, however, most of the business cases and use cases have been isolated from the real world [Med]. Due to this business cases that require legal compliance and / or strong authentication and identification have become particularly difficult to implement on the BlockChain. Some of these include online banks, online rental service, and many more. In the same vein, there are a number of real world applications such as traditional banks, insurance, immigration, etc. that could benefit from the advantages of blockchain but due to regulation, compliance, and strong authentication requirement, these use cases are walled off outside the blockchain.

Apart from this, on Therefore there is a clear demonstrated need for a bridge in the gap between the blockchain world and the traditional (mundane) world.

3.2 Who feels this problem the most?

The banking sector currently faces this problem the most when it comes to monitoring user activities. Banks need to prevent terrorist activities by constantly monitoring their customers and making sure they comply with anti-money laundering regulations. This typically involves rigorous background checks on transactions and the parties involved. This is the "so called" Know-Your-Customer process (KYC) . The entire process may take up to 2 weeks depending on the situation. The result is delayed transactions and increase in

cost, long labor hours, and high fines if regulations are not met.

Apart from the cost in time and money, monitoring customers also come with privacy concerns especially in the EU with the existence of the GDPR. This has created a seemingly insurmountable fallacy of *GDPR* compliance versus Banking regulation compliance for banks in the EU region. Thus they are required by law to monitor customers, however, monitoring customers may cause privacy breaches, resulting in possible loss of customers.

In summary, the major problem is the high costs and delay in KYC processes faced by banks especially in Europe.

3.3 How will a solution be used?

For a full self-sovereign identity, we propose a hybrid solution[1]. This solution will be used by the 3 stakeholders discussed in Chapter 2.

Authentication agent will be provided with a blockchain gateway that extends traditional authentication to the blockchain. The blockchain gateway is equivalent to the conventional blockchain oracle (reference). This gateway will be responsible for creating verifiable claims for users (Subject) as well as creating the infrastructure necessary to verify claims whether by a Subject itself or an entity approved by Subject. Authentication agents serve as the main trust anchors (identity providers) in the solution.

Authorization agents (service providers) are provided with an application interface that can initiate specific claim verification of users, as well as trigger notifications of events related to specific claims.

Subject will be provided with a mobile app that implements a BlockChain wallet to securely store their own claims generated from various Identity providers. Subject may decide which service providers to share their claims with and how to share those claims.

All 3 applications;

- BlockChain gateway for Authentication agent
- Application Interface for Authorization agent, Subject interface with the BlockChain backend
- Mobile wallet application for Subject

act as one solution made up of just a BlockChain backend and a BlockChain gateway. Thus just as a mobile app interfaces with a database backend and web applications, the

[1]We define a hybrid solution in this context as a solution that merges both off-chain and on-chain approaches.

inner workings of this solution should be abstracted and transparent to the entities using it.

Furthermore, the solutions may be provided as APIs to enhance easy integration into existing systems. Lastly, must be implement in such a way that these 3 stakeholders operate independently of each other.

If the Hybrid solution is used this way, it will provide the following core advantages:

1. KYC processes can be achieved in a matter of seconds, thus, solving the issue of high costs and delayed time.
2. Subjects also have a full blown self-sovereign Identity, which increases user experience and maintains GDPR compliance.
3. The blockchain transactions performed by the bank on users during the KYC processes constitute a KYC proof. This proof may be generated in such a way that it is reusable by other service providers. This may be an alternate source of income for banks.
4. The efficiency problems, as well as Type 1, Type 2, and Type 3 attacks Section 2.1.3 are minimized.
5. The blockchain transaction on Claims, even though they provide minimal information on Subjects can be trust enough (tamper proof audit trail) for external auditors and compliance.

Chapter 4

Methodology

4.1 Available methods

The main objective of this research is establish a middle ground between centralized identities, which are characterized by strong identification and easy legal compliance, and decentralized identities which allow for anonymous transactions, higher efficiency, and security. To achieve this major goal, 5 research questions were created:

1. How do we securely store, update and retrieve trusted identity data from the blockchain?
2. What are the minimum security requirements needed to enforce secure granular data management on the blockchain
3. To what extent is the chosen method in 1 above scalable?
4. Are there any other use cases for such a solution in 1 above?
5. How do we achieve compliance in use cases that require strong authentication (Example Governmental level, Banking)?

There were two main ways to gather the necessary information and analyze them to come to a conclusion. The first option was to use empirical analytical methods (objective methods), where, for instance a survey is given to a large group of experts in the field of Identity Management and blockchain. However, apart from the fact that there would not be enough experts in such a study to obtain an objective results, the expected solution to the 5 problems listed above are quite subjective.

Therefore, there was a need to gather information and analyze them using more interpretative group of methods. This subjective method allowed us to focus on understanding Identity Management problems and blockchain in a more comprehensive, holistic way.

4.2 Chosen method and Process Report

Our chosen method proceeded in 4 phases: Preparation phase, Design, Expert verification, Cryptographic verification.

4.2.1 Preparation phase

The Literature review phase focused on gathering information about the state-of-the-art in Digital Identity Management, blockchain, and Self-Sovereign / Decentralized Identities.

To better understand Digital Identity Management, this research was conducted at E-Group in Budapest. This company is responsible for managing the eIDAS network node in Hungary, has an existing identity management product line and is developing the Hungarian node for the PEPS (Pan-European Proxy Service).

What is eIDAS? [Digd] The Connecting Europe Foundation electronic Identification, CEF eID [Digd], is a set of services (including software, documentation, training and support) provided by the European Commission and endorsed by the Member States, which helps public administrations and private Service Providers to extend the use of their online services to citizens from other European countries. This is realized through the mutual recognition of national electronic identification (eID) schemes (including smart cards, mobile and log-in), allowing citizens of one European country to use their national eIDs to securely access online services provided in other European countries. The goal of CEF eID is to meet the eIDAS regulation by 29th September, 2018. [Digd] [Digc]

This is achieved through a sophisticated pan European network consisting of f a number of interconnected eIDAS nodes, one per participating country, which can either request or provide cross-border authentication.

The eIDAS network was closely studied because, it is one of the most sophisticated blends of federated and user-centric identity Management systems today, hence, gathering information from this network provides a keen insight into Identity Management Systems in general.

In terms of blockchain, Bitcoin, Ethereum, and HyperLedger were closely examined through the published white papers, yellow papers, and beige paper. Sample smart contract codes were also written and tested on the Rinkeby network to fully understand the strength and possible weaknesses of Ethereum. Bitcoin and hyperledger were also analyzed through similar practical experiment.

Lastly to better understand Self-sovereign identities, publications on the Ethereum ERC 725, Sovirin, and the Estonian Identity Management System. In house documentations were also studied from TU Berlin's Identity Chain project. All researched literature from

the literature review phase have been provided in the bibliography.

4.2.2 Design phase

One major roadblock encountered during this research was the scarcity of information / literature concerning standardized ways of managing sensitive information (such as Identity data) on the blockchain. To overcome this hurdle, the design phase was created iteratively propose secure and privacy preserving ways of achieving blockchain identities. The proposed designs were subjected to expert analysis by ICT team at Egroup, Professors at ELTE, as well as experienced business personnel in the area of Identity Management such as OTP bank, Posta Italiane, and other business partners.

4.2.3 Expert verification phase

In the expert verification phase the various proposals in the design phase were analyzed by the experts mentioned above in 2 hour meetings in person or over Skype. The results of each iteration were documented taking note of emerging patterns.

4.2.4 Cryptographic verification phase

All proposals that survived the expert verification phase were reduced to the corresponding cryptographic primitives such as hash functions, encryption / decryption, signatures, etc. If any supporting cryptographic foundations are found, the proposal is documented pending the results section. After about 5 iterations of these 4 phases the results of each phase were distilled into the results section.

4.3 Actors, Activities and Activity Diagram

This chapter more formally describes all the direct and indirect participants of the research (actors), the activities each performed, and how these activities translated into the final results.

4.3.1 Actors

There were 5 main entities involved directly or indirectly in this research:

1. Research Student / Author (Kwadjo Nyante)
2. Supervisor 1 (Industrial Supervisor at E-Group
3. Supervisor 2 (Academic Supervisor at entry University - University of Twente)
4. Supervisor 3 (Academic Supervisor at exit year University - Eotvos Lorand University)
5. strategic partners (OTP Bank, Posta Itialiane, IdentityChain team TU Berlin)

4.3.2 Activities

Table 4.1 and Table 4.2 give a full description of all the atomic activities performed throughout the research. and Figure 4.1

Activity	Description
A1	Gather information about the state-of-the-art in identity Management ad blockchain.
A2	Propose a possible solution for the problem section that is provably secure and technically feasible.
A3	Evaluate if a given solution is technically feasible and cryptographically secure.
A4	Evaluate if a given solution has business feasibility and actually addresses the business needs
A5	Make suggestions
A6	Add output of the iteration to the input of A1

Table 4.1: Activities and their Description

Activity	Input	Output
A1	technical papers, books, lectures, videos of seminars, similar solutions, and interviews	A pattern explaining the causes of Identity Management problems, and exactly what requirements an ideal solution should have.
A2	Output of A1	A description of a possible solution and the supporting security and cryptographic proofs.
A3	Output of A2	Yes or No
A4	Output of A2	Yes or No
A5	Output of A2	Answer if Yes or No, why and what can be added?

Table 4.2: Activity Input and Output

A3 and A4 both take as input the output of A2 to avoid missing possible solutions just because they were rated as not technically feasible. This could easily happen if the input to A4 is the output of A3. Thus in summary, technical evaluation (A3) and business evaluation (A4) are independent of each other.

4.3.3 Activity Diagram

fig. 4.1 shows the activity activity diagram of the entire research. It should be noted that the research progressed through several 4 iterations. At the end of the four iterations:

- the pattern developed as the output for A1 formed the foundation for the results (claims, trusts, digital identities)
- the output of A3, A4, and A5 formed the main results
- the intermediate solution at the end of the first, second, and third iteration formed the discussion section of the results and informed the answering of the research questions

It should be noted that activities A1, A2, and A6 are performed by Author. A3 and A5 are performed by Supervisor 1, Supervisor 2, and Supervisor 3. A4 and A5 are performed by the strategic partners of E-Group.

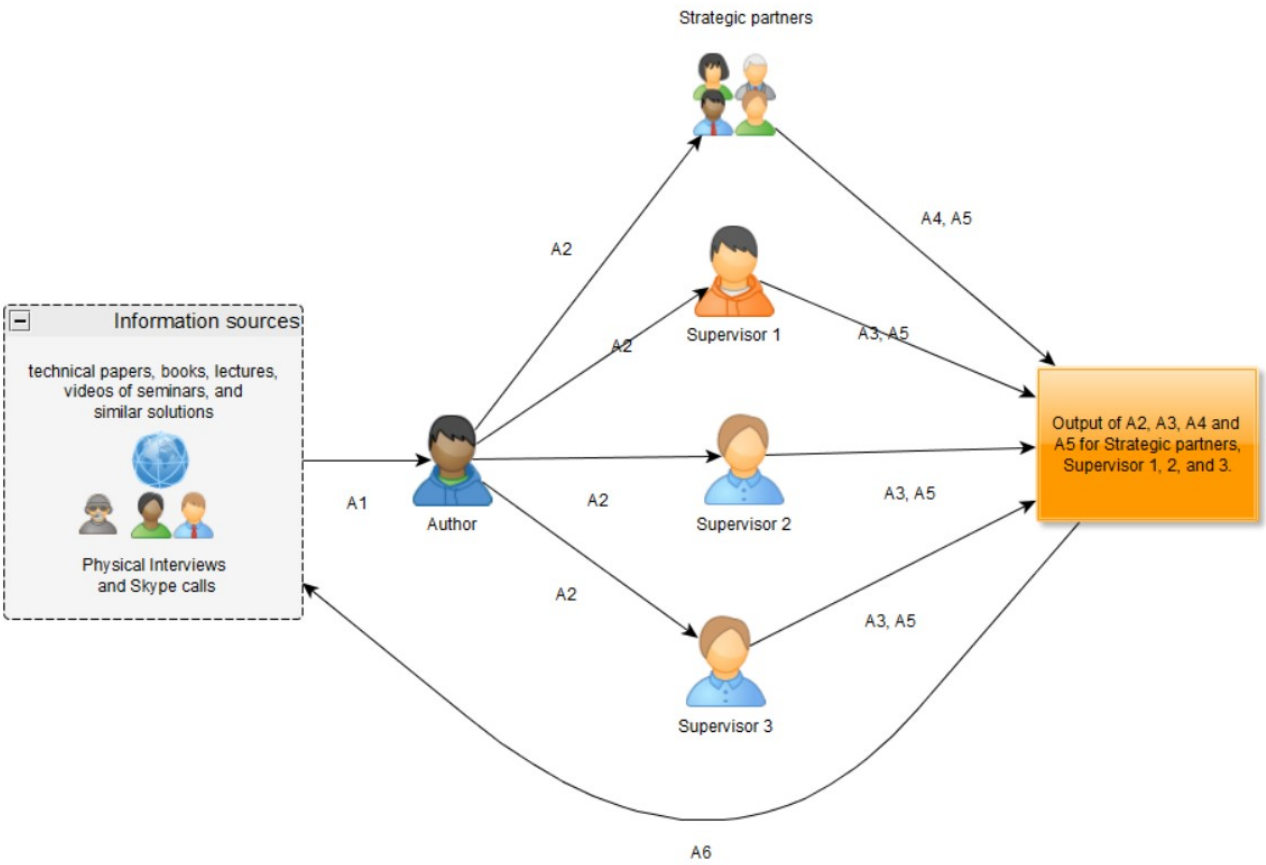

Figure 4.1: Activity diagram showing a single phase of the research

Chapter 5

Results

Identity Management Schemes in general were reduced to the atomic building blocks. Thus, having a clear idea of the minimum building blocks / requirements that make an Identity Management Scheme, these building blocks can be redesigned to eliminate the problems specified in Chapter 2 (Efficiency and safety problems, Type 1, Type 2, and Type 3 attacks). Additionally, by redesigning these building blocks using the blockchain, the result / solution also inherits the advantages of the blockchain.

It is important to note that this paper does not delve into the implementation details of a self-sovereign Identity Management scheme, however, it serves as a general guide (whitepaper) for conceptualizing, developing, and evaluating the security of any Identity Management Scheme (especially Self-Sovereign Identity Management Systems). We leave the actual implementation details to the designer, thus, giving them a wide discretion to tailor the chosen Identity Management System to their business needs.

5.1 Formal Definitions of Identity Management

As seen in Chapter 2, there are many configurations of Identity Management, each with their own pros and cons, therefore, it is difficult to create one standard formal definition of Identity Management. Nevertheless, for the purposes of this research, it is crucial to formally and atomically define the core parameters in Identity Management, because it forms the foundation for the security proofs shown in Section 5.3.

5.1.1 Subject Data

Definition: *Let Subject Data be represented by any subset of an infinitely large space* $D = \{d_0, d_1, d_2, \ldots, d_\infty\}$ *where* $d_i \in \{0,1\}^n$ *is an atomic attribute of Subject.*

Atomic attributes of Subject include all possible attributes of individuals and organizations such as: names, ages, genders, eye colors, tax numbers, and many more. This forms an infinite pool of all possible attributes (structured or unstructured) Subject can have as shown in Figure 5.1. These

attributes, when matched to (claimed by) a Subject, becomes an attribute belonging that Subject.

Figure 5.1: Unclaimed pool

5.1.2 Claim

Definition: *Let a Claim be represented by any member of an infinitely large space* $C = \{c_0, c_1, c_2, \ldots, c_\infty\}$ *such that* c_i *is a reference to a logical link between two different Subject data elements* d_i *and* $d_j \in D$*, where* $c_i \in \{0,1\}^n$ *and* $d_i, d_j \in \{0,1\}^n$

For every claim in space C,
$\exists\ T, F : T \subset C$ *and* $F \subset C$*, such that* $T \cup F = C$ *and* $T \cap F = \varnothing$ *where* $T = \{c_{t1}, c_{t2}, \ldots, c_{t\infty}\}$ *and* $F = \{c_{f1}, c_{f2}, \ldots, c_{f\infty}\}$*, where T is the set of all possible true claims and F is the set of all possible false claims.*

$\exists\ V : \forall\ c_i, V(c_i) = \{0,1\}$ *where V is a polynomial time algorithm (efficient algorithm) such that*

$$V(c_i) = \begin{cases} c_i \in T, & 1 \text{ (Claim is true)} \\ c_i \in F, & 0 \text{ (Claim is false)} \\ otherwise, & 0\ (c_i \notin C) \end{cases}$$

Every element belongs in set C belongs to either T (true claim) or F (false claim), where T and F are mutually exclusive and there is not element in set C that belong both T and F. The verification function output 1 is clam is true and 0 if claim is false. The integrity of a claim depends on how securely V is implemented.

Figure 5.2: Claimed pool

In Figure 5.2 above, Bob, Alice Ken, and Oli have made some truthful claims on some data from the unclaimed Subject data pool. Bob claims to 18 years old, Alice claims to be Single, 22 years old and from the USA, etc. Eve has made a false claim of being both male and female. The verification function V should be able to easily distinguish between a true (valid) claim as Bob, Oli, Ken, and Alice have made, and a false (invalid) claim as Eve's claim.

5.1.1. Lemma. *A digital identity is a verified Claim.*

5.1.3 Strong Claim vs Weak Claim

Definition: *Let a strong claim c_s be a reference to a logical link between Subject data d_i and $d_j \in D$, such that either d_i or d_j is globally unique, unforgeable, persistent and physically verifiable using a polynomial time algorithm such that $V_s(c_s) = \{0, 1\}$.*
Such a V_s is called a strong verification function.

An atomic example of a strong claim is a claim that links a person's name with their DNA results, facial structure, fingerprint, retina, or any other globally unique and physically verifiable attribute. Such a claim can be easily verified with a strong verification function Vs such as DNA test, facial recognition, fingerprint scan, retina scan, etc. that matches the claim.

Definition: *Let a weak claim c_w be a reference to a logical link between Subject data d_i and $d_j \in D$, such that both d_i and d_j are not globally unique and (or) physically verifiable.*

For verification, such a claim uses a polynomial time algorithm V_w such that $V_w(c_w, x) = \{0, 1\}$, where x is some secret information. Such a V_w is called a weak verification function.

An atomic example of a weak claim is a claim that links a person's name with their email, password, messages, phone number, age, gender, insurance number, and height. Such a claim is verified with a weak verification function V_w such as email - password verification. Note although age, gender, height and the other fields are physically verifiable, they are not globally unique, unforgeable, or persistent and hence a verification function.

5.1.4 Trust and Digital Identity

Trust w.r.t Digital Identities is a critical component of Identity Management Systems that needs to be formally defined.

Definition: *Trust, t_i, can be atomically defined as an intangible component generated as the by product of the sharing of a claim $c_i \in C$ between two parties A and B. This intangible component ti allows further interactions of a particular nature between A and B, eliminating the need for both parties to share further claims.*

A trivial example is sharing a claim that you can visual read a blurry representation of the word "security" on a computer. The claim, therefore, becomes **Bob** $\rightarrow$ **security**. If this claim has a secure and efficient verification function, it may be used as an online proof that you are not a robot, hence eliminating the need for further authentications of this nature throughout the interaction.

5.1.2. Lemma. *In general, trust is generated when a secret is shared between two parties.*

We argue that a claim constitutes a secret, therefore when any claim is shared, whether valid (true) or invalid (false), it has the consequence of generating valid trust or invalid (misplaced) trust respectively. The degree of sensitivity of the claims dictates the degree of trust. Lastly, the lifetime of trust t_i is directly proportional to the amount of time, t, when the shared claim c_i remains valid under a particular verification function V.

5.1.3. Lemma. *Atomically, establishing a digital identity establishes trust between two parties; the owner of the Identity and the Verifier of the identity.*

This lemma is not invertible because trust can be established without first establishing a digital identity. For e.g. trust can be established when one party performs a sacrifice, trust can be established by a gut feeling, or one person may trust another person just because of how they smell. However, we strictly do NOT consider this type of trust generation as a

means of establishing a digital identity because at the time of writing, generation processes involving sacrifice, gut feelings, scent, etc. cannot be mathematically modeled.

5.1.4. Lemma. *Trust is transitive.*

Generally, a person can trust you because they can personally verify your identity. We argue that a person can also trust you because someone else they trust can verify your identity. If A trusts B, and B trusts C, then A can trust C to some degree. At least somebody trusts you so I can trust you. This nature of trust is called transitivity.

5.1.5 Transforming weak claim into strong claim

In accordance with Lemma 4, a weak claim may be easily transformed into a strong claim by simply chaining them (weak claim and strong claim) together. Chaining in this context is defined as simultaneously making two or more such that, verifying any one claim requires verifying at least one strong claim. The figure below shows how Bob may strengthening an arbitrary set of weak claims using chaining.

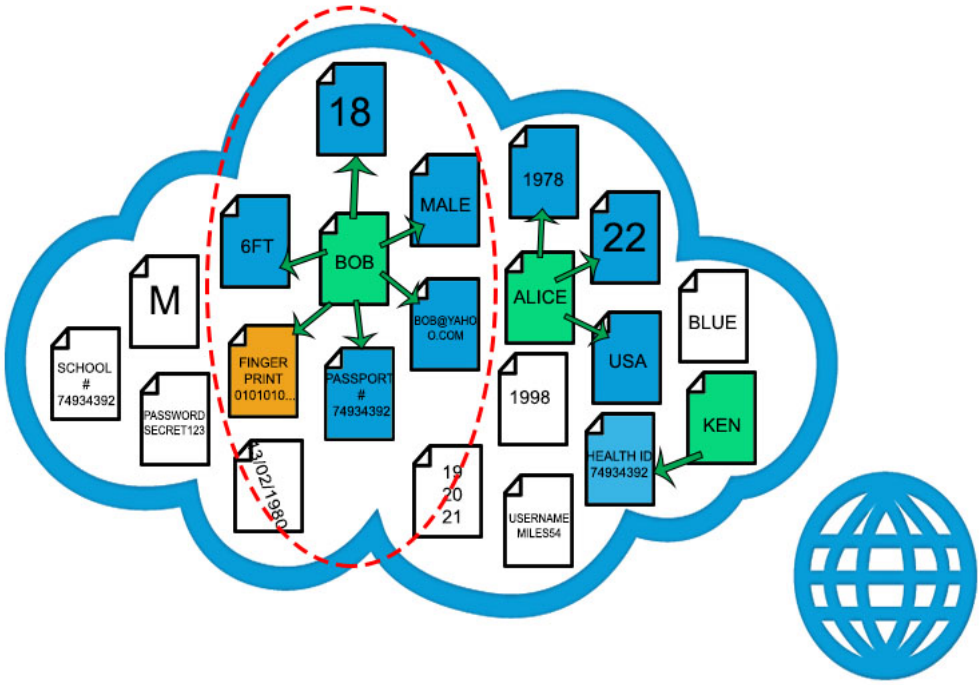

Figure 5.3: Transforming weak claims into strong claims

In the Figure 5.3 above, Bob has 5 weak claims shown in blue. The weak claims are his age (18), sex (male), email (bob@yahoo.com), height (6ft), passport number (74934392). Bob then makes one strong claim (fingerprint) shown in orange. If verifying any one of these claims requires verifying at least one strong claim, then all the claims (including the weak claims) may be considered as equivalent to a strong claim as shown below in Figure 5.4

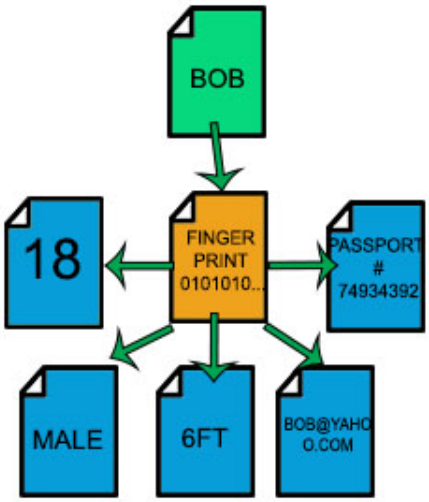

Figure 5.4: Chained claims

It should be noted that if all five weak claims (shown in blue above) are to be verified, one strong claim verification is enough in addition to the five weak claim verifications. As a rule of thumb, in a chaining method at least 2 claims / verifications must be made. One strong claim and the subsequent weak claims. One may also decide to use multiple strong claims for enhanced security, however, in most use cases it may also be necessary to consider efficiency.

5.1.5. Lemma. *A chain of claims needs at least one strong claim to transform other weak claims into strong ones.*

Proof: The proof for the above is quite trivial. The formed chain is unforgeable and globally unique since the strong claim in the chain is also unforgeable and globally unique. Also, verifying the chain of claims requires physically verifying the strong claim in the chain. Therefore we can conclude that a chain of claims with at least one strong claim is unforgeable, globally unique, and requires physical definition. This is in line with the definition of a strong claim, hence proven.

5.1.6 Identity Management System

Definition: *An identity management system at the core consists of a triplet (3) of randomized polynomial time algorithms (efficient algorithms) G (Generate), T (Transform) and V (Verify) such that:*

$G : (m, 1^n) \rightarrow (c[], x)$,

claim generation algorithm that generates an array $c[] \subset C$ consisting of an arbitrary

(m) number of claims, where c[] = $\{c_1, c_2, c_3, ..., c_n\}$, and $x \in \{0,1\}^n$ is some secret information known only to / in possession of the Subject and is used to prove that subject owns the claim.
E.g. this could be a password used to protect claims like email, date of birth, insurance number, etc when registering on a website.

$T : T(c[], y) \rightarrow (c'[], k)$,

Transform algorithm that performs a cryptographic transformation on generated set of claims c[] using $y \in \{0,1\}^n$, where y is some secret information known only to the identity provider. c'[] is the transformed output and $k \in \{0,1\}^n$, is information necessary for verifying claims.
E.g the case of a website like Facebook, the Transform algorithm takes all personal data (claims) belonging to a person, c[], and securely stores these claims in their database. For secure storage, the Transform algorithm uses an encryption transformation with secret key y. The result is c'[] securely stored and unintelligible to all external parties apart from the Identity Provider (Facebook) and the Subject (Facebook user). The Subject may then use k, which is say the Facebook username, and the verify function to access their claims.

$V : V(c'[], x, k) \rightarrow \{0,1\}$,

verify algorithm allows external parties (other than identity provider) to verify the claims. With the exception of a mandatory field k, c'[] and x are optional inputs to the verify algorithm. The output 1 corresponds to successful verification and 0 corresponds to an unsuccessful verification.
E.g, in the same Facebook scenario, the Verify algorithm is invoked with k = username, and x = password. If, and only if, the Verify algorithm returns a 1, the user is granted access to their account (verified claims). The user account or verified claims here is equivalent to the Subject's digital identity, in accordance to Lemma 5.1.1.

5.2 Configurations of Identity Management Systems

We argue, that the type of Identity Management system depends on the configuration of the Generate, Lock, and Verify algorithms as formally defined above. In other words, whether an Identity Management is centralized, federated, user-centric, or self-sovereign depends on three things:

1. How claims are created (Generate algorithm)
2. How claims are stored (Transform algorithm)
3. How verification is done (Verify algorithm)

5.2.1 Claim Generation

There are two main configurations of claim generation. This depends on who generates the claims. In general, a claim may be generated by the owner of those claims (Subject), or an external Identity Provider (Authentication Agent).

If the claim is generated by Subject the Generate algorithm is trivial to implement since the Subject has access to the claims, and is also the creator of the secret information x. On the other hand, if a claim is generated by an external Identity provider, special steps have to be taken to protect the secret x from falling into the hands of anyone else apart from the Subject. Not even the Identity provider may have access to access.

Practically, this can be achieved by storing the secrets x (e.g. password) as salted hashes that are both CPU and memory hard to compute. This makes it computationally difficult for even the Identity provider to learn secret x, but very easy for Subject in possession of x to access the claims c[].

5.2.2 Claim Storage

In a quite similar fashion, there are two main configurations for claim storage. A claim may either be stored by Subject or the external Identity provider who generated the claims.

Depending on the kind of storage, the Transform algorithm has two configurations, thus it may be an encryption transformation or signature transformation depending on the entity that stores the claims.

1. Identity Provider storage: If claims are stored by Identity provider (Claim generator), the output c'[] of the Lock function is only intelligible to the Identity Provider, therefore, algorithm T is an encryption transformation. No third party other than Subject and Identity Provider may have access to the claims unless allowed by Subject. Subject may have access to their claims by using k and their secret information x in the verification function.

2. Subject storage: If claims are stored by Subject, then the output c'[] is intelligible only to Identity Provider and Subject, however, a mark is left on the c'[] to prove that it was created by the Identity provider. This is essentially a signature algorithm were c[] is the data to be signed, y is the private key of the identity provider and c'[] is the signature generated. In this case, k becomes the public key of the identity provider and can be used in combination with c'[] in the verification function.

5.2.3 Claim Verification and Signature Verification Paradox

How claim verification is done is one of the most crucial determining factors of the type of Identity Management System. There are two types of claim verification: Verification by owner of the claims (Subject), and verification by external entity (usually Authorization

agent). If verification is done by Subject, the only pitfall to be concerned about is that the secret x is not easily guessed or somehow revealed during verification. This is trivial to implement, e.g. forcing a particular character set and length for passwords.

On the other hand, if a claim is to be verified by a third party (an entity which is NOT the Subject or Identity Provider), it typically involves sharing of claims. This is because the Verification function needs both c'[] (a signed array of claims which is typically a signed hash), the original claims c[], and k (the public key of signer (Identity Provider)). No encryption or cryptographic transformation can be performed on the claims because that will result in an unsuccessful verification.

In effect, this creates a paradox we have termed the **"Signature Verification Paradox"**. Thus verifying a signature is crucial for third parties (Authorization agents), however, it requires the Subject to reveal possibly sensitive information which may compromise privacy.

Furthermore in accordance with Lemma 5.1.2, a claim constitutes a secret, therefore, its sharing results in the generation of trust. This trust generated allows authorization agents to perform just one invocation of the Verify algorithm when provided resources or services. However, as formally defined in Section 5.1.4, it is clear that trust is delicate and therefore should be managed carefully. It is important to note that if trust is centralized, thus all claims and verifications are issued and performed respectively by the same Authentication agent, this single agent accumulates an undue amount of trust effectively transforming the whole scheme into a centralized or federated identity scheme. The following are some three ways in which verification can be performed with a third party (Authorization agent):

1. **Full Disclosure**: Subject can provide the Authorization agent with a signature of the claims, the original claims (c[]), and the public key (k) of the Identity Provider. This would result in a very simple Verify algorithm, which is easy to implement and efficient in terms of speed and computational resources. On the other hand Subject loses their control over the claims immediately it is disclosed and thus have no transparency of how the claims are used. If Authorization agent is benevolent, claims will be used for their predefined purpose, otherwise it may result in a Type 2 attack as established in Section 2.1.3.

2. **Anonymized Disclosure**: If a third party (Authorization) agent has to verify the address and salary of a Subject, instead of exposing their actual address / salary, the Subject may use any anonymization technique that provides say their city and their salary range. This is called an anonymized disclosure. This can be done in several ways: Getting anonymous claims from the Identity Provider, employing a third party that performs anonymous verification using the Verify algorithm, using Zero Knowledge Range Proof (ZKRP) [KRvW], etc. Whichever technique is used, it should be noted that anonymized disclosure only provides a degree of control and transparency for Subject's claims and may be susceptible to whole classes of

anonymization attacks such as record linkage attacks, homogeneous attacks, etc. [2]. However it provide some sort of balance between efficiency and privacy.

3. **Zero Knowledge Disclosure**: It is possible to implement the Verify algorithm in a special way such that a Subject may prove their claims to a third party without revealing anything about the claims. Theoretically, this can easily be achieved with Zero Knowledge Proofs (ZKPs) [S$^+$03], Bullet Proof [BBB$^+$], Zero Knowledge Succinct Non-Interactive Arguments of Knowledge (ZKSNARK)[S$^+$03], Sigma Protocols[S$^+$03], etc. This provides the ultimate control and transparency, thus privacy, for Subject, however, in practice implementing such a Verify algorithm securely is notoriously difficult and prohibitively computationally expensive.

5.2.1. Lemma. *Identity Management is equivalent to trust management.*

5.2.4 Comparison of different Identity Management Systems

From the above possible configurations configurations of Identity Management Systems algorithms (G,T, and V), we provide the following table that correlates the choices of configuration with the type of Identity Management System.

	Generate	Transform	Verify
Centralized	Claims are generated by Identity Provider	Claims are stored by Identity Provider	No transparency, claims are controlled by Identity Provider
Federated	Claims are generated by multiple Identity Providers	Claims are stored by the respective Identity Providers	No transparency, claims are controlled by multiple centralized authorities
User-Centric	Claims may be generated by user but practically easier to let Identity Providers generate them	Claims are stored by the Identity Provider	Verification results in full disclosure of claims to service providers (Authorization agents)
Decentralized	Claims may be generated by users or Identity Providers	Claims are usually stored by Subject (users)	Verification may leak information about Subject, thus, full disclosure or weakly anonymized disclosure
Self Sovereign	Claims may be generated by users or Identity Provides	Claims are strictly stored / controlled by user	Verification is strictly either Zero knowledge Disclosure or very strongly anonymized disclosure

Table 5.1: Correlating formal definitions with evolution of Identity Management Systems

5.2.5 Self-Sovereign Identity Management System

Since the solution to the research problems is a Self-Sovereign Identity Management System, it is necessary to formally define a Self-Sovereign Identity Management System.

Definition: *We formally define a self-Sovereign Identity Management system as a triple (3) of specialized randomized polynomial time algorithms (efficient algorithms): G (Generate), T (Transform) and V (Verify) as defined in section 5.1.5 (Identity Management Systems), such that:*

G is invoked by the user whether or not claims are generated by Subject or Identity Provider,
T is strictly a signature transformation where Subject stores the claim(s), and
V is invoked in such a way that it only outputs 0,1 to all parties and no other information.

The implementation of these properties creates such an Identity Management System that:

- Any entity (user (Subject) or Identity Provider (Authentication agent)) may create claims
- Every entity is responsible for the storage and security of their own claims
- Verification should not deprive users of their control, oversight, or privacy over their claims (data).
- There should NOT be one single source of trust for all verification or claim generation, thus anyone, even regular users (Subjects) may be an Identity provider for claims, thus responsible for signing other users' claims.

This essentially decentralizes the trust, and transforms the identity management system into a reputation system giving the user the choice to determine the amount of trust to place in any Identity Provider based on its reputation.

5.2.2. Lemma. *A Self-Sovereign Identity management system is a specialized Secure Digital Signature scheme.*

We argue that a Self-Sovereign Identity is a specialized Secure Digital Signature with claims as the data to be signed and a special verification function that reveals zero knowledge about the claims to the verifier.

5.3 Formal Security Requirements of a Self-Sovereign Identity Management System

In accordance with Lemma 5.2.2, we construct a security game for Self-Sovereign Identity Management systems. The security game is made up of two stages: the first stage for the Transformation algorithm (oracle) T, and the second stage for the Verification oracle V. There security requirements for the Generation oracle, G, is trivial, therefore, it is not shown in this paper. One requirement to consider when implement G is that the secret x should be chosen from a sufficiently large space in order to thwart a rational resource bound adversary from iterating through the secret space (brute-force).

5.3.1 Transformation Oracle Security

In accordance to Lemma 5.2.2, the Transformation algorithm of a self-sovereign Identity management system can be reduced to a simple Secure Digital Signature Scheme. Therefore, the security game shown here is very similar to the proofs used in Secure Digital

Signatures [S$^+$03].

Definition: *A Self-Sovereign Identity Scheme (G,T,V) is (t, q, ε)-secure, if for arbitrary efficient adversary Z^T with time limit t and number of requests q to signature oracle T, the probability that Z^T outputs a valid [claim, signature] pair [$c^f[], c^{f'}[]$] for a new claim is at most ε, i.e.*

$$\underset{(c[],c'[],k)\leftarrow(G,T)}{P}[V_k(c^f[], c^{f'}[]) = 1, where(c^f[], c^{f'}[]) \leftarrow Z^T(k)] \leq \varepsilon \tag{5.1}$$

or in more compact form

$$\underset{(c[],c'[],k)\leftarrow(G,T)}{P}[V_{pk}(Z^T(k)) = 1] \leq \varepsilon \tag{5.2}$$

In summary, if we gave a polynomial time bound adversary a finite amount of arbitrary queries to the Transformation oracle (T), the probability of forging a claim,signature pair that outputs a successful verification from the Verification algorithm (V) should be less than $\varepsilon = 1/polynomial(|k|)$. This is quite similar to EU-CMA (Existential Unforgeability Chosen Message Attack). We have termed this **Existential Unforgeability Chosen Claim Attack (EU-CCA).**

5.3.2 Verification Oracle Security

Due to the strict nature of the Verification oracle (V) of a self-sovereign Identity Management System as defined in Section 5.2.5, the security of the verification oracle is quite similar to the definition of semantic security. Like the information theoretical analogue of perfect encryption, this is perfect verification (Zero Knowledge Disclosure).

Definition: *A Self-Sovereign Identity Scheme (G,T,V) is t(n), ε(n) semantically secure if given an arbitrary distribution C of claims, arbitrary partial information function g, and arbitrary efficient adversary Z^V with resource bound t(n), there exists another efficient adversary Z'^V with the same resource bound such that:*

$$\underset{\substack{c\leftarrow C\\(c[],c'[],k)\leftarrow(G,T)}}{P}[Z^V(1^n, V(c[], c'[], k), k) = g(m)] \quad - \underset{\substack{c\leftarrow C\\(c[],c'[],k)\leftarrow(G,T)}}{P}[Z'^V(1^n, V(c'[], k), k) = g(m)] \leq \varepsilon(n) \tag{5.3}$$

A partial information function: g: C: $\rightarrow \{0,1\}^*$ (C claim space)
E.g: $g(c[]) = 1, c[] \in \{0,1\}^n$, if $c[]$ has odd parity

In summary, given two polynomial time bound adversaries to the verification oracle:

- one (Z^V) with access to the Generation algorithm, Verification algorithm (containing original claims, signature, and public key k), and public key k, and

- the other ($Z^{'V}$) with access to the Generation algorithm, Verification algorithm (containing only the signature and public key), and public key,

the probability difference of gathering partial information between the two adversaries Z^V and $Z^{'V}$ is negligible, thus, $\leq \varepsilon$, then the Self-Sovereign scheme is semantically secure. *In other words, in Verification, knowing or not knowing the original claims (c[]) provides no information advantage to any efficient adversary.* ***Thus, to be semantically secure, verification should be performed without revealing the original claims.***

5.4 Blockchain Gateway

In our design, the blockchain gateway *(also known as blockchain oracle)* acts a a major Authentication agent and linking pin between the blockchain world and off-chain world. With this particular self-sovereign Identity scheme, the blockchain gateway in essence acts as an entity that implements the Transformation algorithm, Verification algorithm, and / or the claim Generation algorithm of the self-sovereign Identity scheme. We leave the low-level implementation details of G,T, and V to the Self-sovereign Identity system designer, however, when designing there is a need to consider efficiency, and reduce any design details to the primitives established in this paper to correlate them with the formal security requirements.

Because the blockchain gateway is a link between the blockchain and off-chain world, it has two main parts working together as one: blockchain presence (e.g a smart contract) and off-chain presence (e.g traditional server). Together, these two entities work as one in a seamless and transparent way to all three stakeholders (Subject, Authentication agent, and Authorization agent). This is why we call it a hybrid solution, because it achieves the balance of the off-chain world and the blockchain world. Figure 5.5 below shows a general design of the blockchain gateway (off-chain presence + on-chain presence) and its major functionalities.

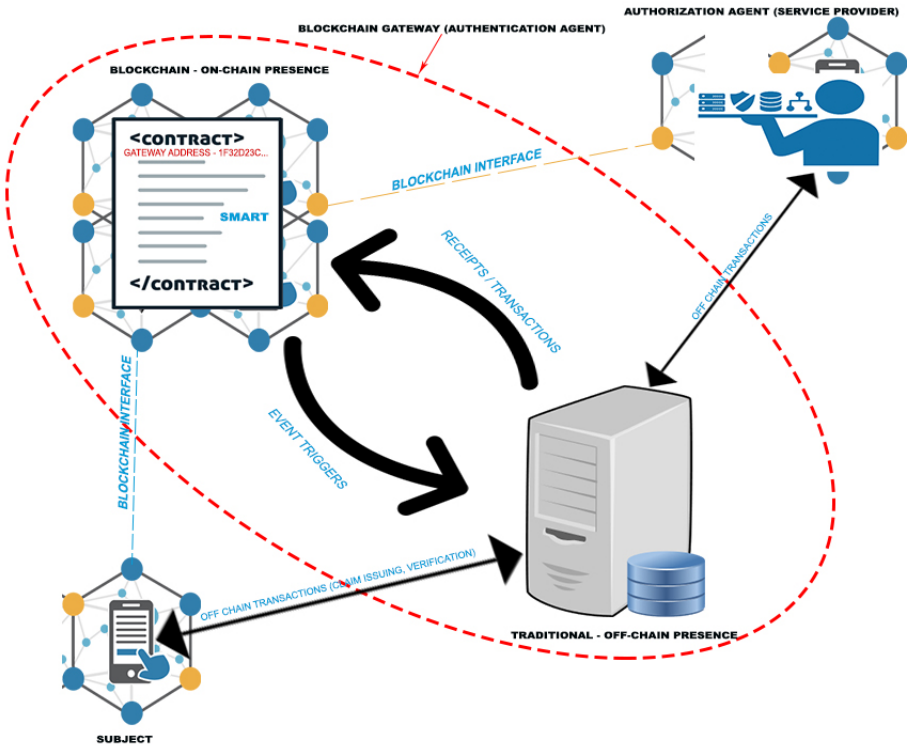

Figure 5.5: Blockchain gateway

5.4.1 Off-chain presence

The off-chain may be a server or a cloud of servers in a trusted environment. In this research, E-Group's eIDAS based Identity and Access Management System may act the off-chain presence. It allows the provision of specific personal and technical attributes depending on the Subject. An off-chain presence performs the following functions:

- **Claim Issuing:** Claim issuing involves the invocation of the Generate algorithm (G) and / or the Transformation algorithm (T). A simple example will be to issue any EU Member State citizen a signed claim of their [Family name, Date of Birth, identification number]. This data is issued over a secure channel to the user preferably on a mobile device. *(Please note that issuing claims directly on a blockchain whether encrypted or unencrypted is strictly undesirable in terms of privacy because these claims automatically become permanent and public indefinitely, thus, providing any adversary all the time in the world to decrypt and misuse claims)*. Claim issuing is easy to implement because most user-centric Identity Management systems already have this infrastructure. In most EU regions this authentication can be done with a citizenship card or some other identity.

In the **Generation algorithm (G)**, the outputs are:
$[Familyname, DateofBirth, identificationnumber] \rightarrow$ **c[]**
private key in the citizenship card or password $\rightarrow$ **x**

In the **Transformation algorithm (T)**, the inputs are outputs are:
$[Familyname, DateofBirth, identificationnumber] \rightarrow$ **c[]**
Identity provider's Private key used for signature $\rightarrow$ **y**
Signed hash of claims $\rightarrow$ **c'[]**
Identity provider's Public key $\rightarrow$ **k**

At the end of claim issuing, the citizen (user) is in possession of the claims($c[]$), signed claims ($c'[]$), and publicly available public key of Identity Provider (k).It should also be possible for a user to send unsigned claims ($c[]$) to the off-chain presence for signing (attestation). In this situation, the Subject has already performed the generation algorithm, therefore, all the off-chain presence has to do is to verify the claims and perform the Transformation algorithm. This is in perfect alignment to the definition of a claim in Section 5.1.3, a claim can only be issued if there is a matching verification algorithm for the claim. This verification algorithm should not be confused with the general Verification oracle of the Identity Management System. In fact, the Verification Oracle (V) described in Section 5.1.6,Section 5.2.5, and Section 5.3.2 is a specialized generalization of the claim verification in Section 5.1.2.

- **Claim Verification:** Claim Verification may be done on chain or off-chain. Full disclosure Off-chain verification involves providing the Authorization agent (Service Provider) with the original unsigned claims and the signed hash of the claims. In this case verification is as simply as using the public key of the Identity provider in a fashion specified by the signature scheme (e.g. ECDSA-256). Anonymized disclosure and some implementations of zero knowledge disclosure may also be used for off-chain verification. It should be noted that it is not advisable to perform full-disclosure verifications on-chain as it may result in leaking private information unto the blockchain. The off-chain presence may also issue pseudonymous identifiers for the Subjects' claims which may be used for on-chain verification.

- **Other Off-chain functionalities:** The off-chain presence may also perform other off-chain functionalities. for example, traditional payment using VISA card by embedding a PSD2 payment interface. This provides more use cases from the blockchain based Identity management.

5.4.2 On-chain presence

The on-chain presence acts as the off-chain presence's proxy of the blockchain. On the blockchain, this presence is simply a public address (e.g. an Ethereum account address). The on-chain presence is strictly supposed to perform the following to functions:

- **Tamper-proof Audit trail:** Quite like the off-chain presence, both Subject and Authorization agent (service providers) also possess arbitrary public addresses. Anytime an Authorization agent wishes to provide some service / resources to a Subject, or anytime a Subject wishes to use some service / resource in possession of an Authorization agent, there is a need to generate trust between these two parties.

  Any of the two parties (Subject or Authorization agent) may initiate the trust generation process (verification of claims). This proceeds as follows:

  1. Authorization agent makes a request from their public address to verify some claims of the Subject. Similarly, a subject may also make a request to be verified in order to access some resource / service.
  2. An event is triggered and the verification process is performed off-chain (full disclosure, anonymized disclosure, or zero-knowledge disclosure) or on-chain using pseudonymous identifiers provided by Identity provider (Authentication agent). Depending on the kind of verification, if an Identity Provider needs to be involved, an event is also sent to trigger the off-chain presence for verification
  3. Verification process outputs a receipt on the blockchain with 1 $\rightarrow$ successful verification or 0 $\rightarrow$ Unsuccessful verification.

  All transactions and receipts involved in the 3-step trust generation or verification process is stored on the block-chain and acts as a tamper-proof audit trail for any compliance or regulatory bodies.

  5.4.1. Lemma. *Regardless of the number or type of claims in any arbitrary verification process, the output is always binary,thus, 0 or 1. For any specific situation, trust is binary.*

  Proof: Between any two agents (e.g. Subject and Authorization agent), a verification process can be explained as a trust generation process to decide whether based on some requirements (e.g age limit, citizenship, etc), enough trust can be generated to share resources or services. We argue that irrespective of the requirements of a verification process (number or type of claims) or the verification algorithm, the result is always a Yes or No, whether you can trust the other party or NOT. Thus, the output is always binary (0 or 1).

- **Smart Contracts:** Apart from the on-chain presence's function of appearing in a tamper proof audit trail, there is a need for some automation of the entire verification process. For example, when do you issue an event or trigger to the on-chain presence?

  This is achieved with a smart contract. For example, on Ethereum, the on-chain presence of the gateway may be a contract account controlled by code that triggers the verification process. This smart contract may accept transactions from any stakeholder (Subject and Authorization agent). It should be noted that in certain

circumstances Subjects or especially authorization agents may want to use a smart contract to automate their processes. *For example, a bank wishes to perform some background check (verify some particular claims) when a transaction exceeds a particular amount. This bank may have their own version of a blockchain gateway which consists of a smart contract and their existing off-chain services. The smart contract can then easily be programmed to trigger the initialization of the verification process.*

5.5 BlockChain Identity Framework

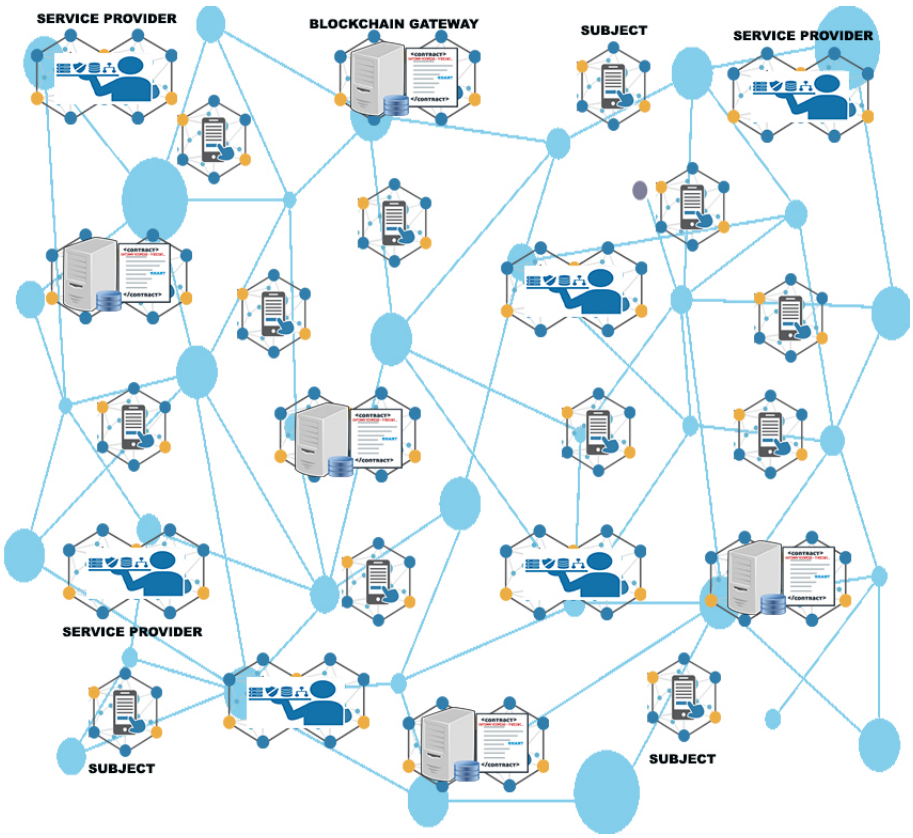

Figure 5.6: Blockchain framework

Figure 5.6 above shows overall design of a self-sovereign blockchain Identity framework. The framework provides a seamless, transparent interaction between off-chain world and the blockchain. It consists of various blockchain gateways (which may also be called Trust anchors, Authentication agents, or Identity providers), service providers (Authorization agents), and Subjects (regular users).

Subjects store their claims in their mobile devices (wallet software) after the claims have been Generated and / or attested (with Transformation algorithm) by one or more of the blockchain gateways (trust anchors). After claim issuing (Generation algorithm) and attes-

tation (Transformation algorithm), the Subject has the choice of which Service providers to share his/her claims with, what parts of a claim to expose, and how long to expose claims to the service provider. This choice is determined by the choice of Verification algorithm used with the service provider and the fact that user consent (in the form of an event trigger) is required whenever a claim is being used. For efficiency, a user can, therefore, program their claim storage wallet to automatically accept all event requests from a particular service provider for a given period of time.

At the same time, a Subject can easily keep track or all transactions involving their claim since they are public on the blockchain, however, because pseudonymous account addresses are used, it provides a high degree of unlinkability / privacy (link blockchain address to a particular user) for the user.

The aim of the Identity framework is to serve as the fabric for generating trust in the network. For some business requirements, less trust is required than others. Thus there may be a need to use more than one blockchain gateway (trust anchor) to verify claims depending on the amount of trust required for a particular action. In general, to be considered as a self-sovereign Identity Management system, the blockchain Identity framework should provide the following main functionalities:

- It should be possible for any user (Subject) to generate their own claims. If a user generates their own claims, it may be verified by any blockchain gateway (identity provider) or even another user providing the Transformation and Verification oracles.
- Blockchain gateways serve as trusted anchors in the framework. Thus in accordance with Lemma 5.1.4, trust is transitive, therefore any service provider may trust a Subject if they trust the blockchain gateway that trusts the Subject.
- A trust anchor may be either a strong trust anchor or a weak trust anchor. A strong trust anchor provides functionality (G,T,V) for interacting with at least one strong claim. On the other hand, a weak trust anchor provides functionality (G,T,V) only for weak claims.
- A claim may be attested and verified by multiple trust anchors.
- In accordance with Section 5.1.5, a verification process may transformed from weak to strong by using at least one strong trust anchor in the verification chain.
- The responsibility of claim storage rests solely on users. Thus users are in charge of both the benefits of being in control of their claims and the associated risks.
- A Subject, by choice, may transfer the risk of storing their claims to a third party or even store it in a protected Distributed Hash Table as backup.

5.6 Discussion

The chosen designs for the blockchain Gateway and blockchain framework were not random. In fact, the rationale behind the design was to closely mimic mundane human identities and how humans verify one another.

An identity may be defined as the sum total of qualities that make up a person or a group of people [MW]. By this definition, these qualities are usually uncountable, and some qualities may be tangible while others intangible. However, the underlying feature is that the sum total of these qualities provide an individual or organization with their uniqueness.

In a similar manner, these qualities can be expressed mathematically as claims. When these claims are digitally verifiable, they may be used as a means of uniquely allocating resources or services because trust has been established (authentication and authorization). Therefore claims and trust are the main building blocks of the solutions presented in this paper.

Once these building blocks were established, the blockchain gateway and blockchain framework were built on top of it. These designs rest on the following requirements and assumptions.

- **Requirements:**
  1. An Existential Unforgeability Chosen Message Attack (EU-CMA) Signature Scheme. This is essential for both the issuing of claims by a blockchain gateway and ownership of accounts or authorization of transactions in the blockchain framework.
  2. A collision resistant, first and second pre-image resistant, and fast hash function. This is essential for the blockchain framework and atomic signing of claims as shown in Section 5.7.2.
  3. A reasonably fast and provably secure consensus algorithm is also essential for designing the blockchain framework.
  4. An efficient Zero-Knowledge Proof or derivative of it with provable completeness and soundness. This requirement is particularly useful when performing Zero-knowledge disclosure verification.
  5. In the construction of the blockchain gateway, sensitive information should not be passed between the on-chain presence and the off-chain presence because all input to smart contracts are public information, so are outputs.
- **Assumptions:** Taking a closer a look at the blockchain framework, there is no reason for a service provider (authorization agent) to trust a Subject when they (the service provider themselves) did not authenticate the Subject. Why should they trust someone they did NOT authenticate? The Subjects claims are provided and / or authenticated by the Identity provider. So how is it that the service provider can trust

the Subject?

The answer to this question is the most crucial and major assumption of both the blockchain gateway and framework. This assumption is that:

1. Trust is transitive. This is in accordance with Lemma 5.1.4. A service provider (authorization agent) may trust a Subject simply because they (the service provider) trusts the Identity Provider of the Subject and the Identity Provider also trusts the same Subject. It should be noted that this trust is NOT equal to unconditional blind reliance or the loose definition of trust. In fact, as formally defined in Section 5.1.4 and Lemma 5.1.2 , it has a particular time-to-live.
2. The random oracle assumption also applies to all one way functions used in the design of the Generation, Transformation and Verification algorithms. These include but are not limited to encryption, signatures, hashes, etc.

5.6.1 Advantages of the design

The design of the blockchain gateway and framework provide the following advantages:

1. **Easy integration:** Most existing Identity Management systems already have existing signature schemes which may easily be used as the Transformation algorithm(T). All that is needed is to build an on-chain presence and claim issuing infrastructure and the IAM will have be transformed fully functional blockchain gateway.

2. **Enhanced interoperability:** One blockchain gateway may interface with several blockchain frameworks. Thus, one Identity provider may be a trusted anchor in several blockchain Identity Management Systems. This also provides a business advantage if service providers are paying for trusted anchors.

3. **User Control:** Users have full control and are the final authority in issues concerning their claims, thus, how claims are stored, verified, and exposed and for how long these actions can be performed.

4. **Inherited blockchain advantages:** The proposed Identity Management scheme also inherits the various advantages of the blockchain such as: enhanced security and privacy, low entry barrier, scalability of trust, zero infrastructure requirement, decentralized consensus, and many more.

5.6.2 Potential Disadvantages

- **Key management:** Key management is one of the major problems this design may potentially face. If Subjects want the freedom to control their own claims, then they also bear the cost, burden and responsibility of secure storage of their claims.

  In order to avoid loss of claims (and for that matter ones identity), we propose

that claims may be encrypted and backed up in any trust secure digital storage. Another thing that can be done is for Subjects to transfer the secure storage risks to organizations willing to protect their claims for a fee. Does this this solution take us back to federated or user-centric identities? The answer is NO if users have a choice about who stores their claims and oversight of any transactions involving their claims (transparency).

- **Scalability:** If this design is to become a global Identity Management framework, the underlying blockchain needs to support millions of transactions per second even if claim issuing is done off-chain and some parts of the Verification algorithm are implemented off-chain. Therefore, there is a need to create or use a blockchain with a very fast consensus algorithm.

- **Distributed Denial of Service Attacks:** If the Verification and Transformation algorithms are not well implemented, it may create a conducive avenue for Denial of Service attacks on blockchain gateways.

  For this potential problem, we argue that a DDOS protection is trivial for the off-chain presence because reasonably secure load balancing feature may be deployed. However, for the on-chain presence, a little effort has to be invested in the underlying blockchain. For example, a good solution is to impose a gas price on transactions as implemented in Ethereum. This serves as an incentive for users to reduce the amount of transactions sent to the on-chain presence, hence reducing the number of trigger events to the off-chain presence.

Thus if designed well, trusting any Subject's claim becomes quite similar to an open and fair voting process, taking into consideration the claim issuer (Identity provider) of the Subject in question and the previous successful / failed transactions on the claims of that Subject.

5.7 Answering Research Questions

Prior to the beginning of this research, 5 questions were posed to provide direction for the research. Answers to these 5 questions have been provided in the following subsections.

5.7.1 How do we securely store, update and retrieve trusted identity data from the blockchain?

Storage: First of all, storing trusted identity information on the blockchain is a non-starter. The blockchain is a public, append-only ledger, therefore, all information stored on it is open to all blockchain participants and hence, theoretically indelible. This means storing trusted identity data like addresses, date of birth, salary, DNA analysis, fingerprint, etc. pose a huge privacy issue as well security issues (especially confidentiality).

Another thing worth mentioning is that any cryptographic transformation (e.g encryption, hashing) of trusted identity data stored on the blockchain is also vulnerable to confidentiality and privacy risks. This is simply because this situation provides a rational polynomial time bound adversary an undue advantage which is time. Such an adversary will have all the time in the world to decrypt or generate the rainbow tables necessary for reversing the hash of the stored identity data, because the stored data is permanent.

In general, depending on the value of the data, storing trusted identity data or any cryptographic transformation of it on the blockchain is at best an unrectifiable mistake, or at worst a horrible security and privacy vulnerability. In the solution presented in this paper, the only information stored the blockchain are transactions and code (smart contracts) related to blockchain gateways, Authorization agents, and / or Subjects. The actual signed claims are stored by the Subject or an entity Subject trusts (if Subject does NOT want the risk/responsibility of storing signed claims).

Update: In accordance with Lemma 5.1.1, an trusted identity data are just verified claims, therefore claims are not updated in the traditional sense but rather new claims are created and attested (using a digital signature). These new verified claims serve as an update to the old claims. For this purpose, we propose for certain claim which are not permanent, such as addresses, etc. a validity period must be attached by the signing Identity provider (Authenticating agent). Permanent claims like date of birth may be signed or attested by the Identity provider indefinitely.

Retrieval: In a decentralized blockchain based Identity, because claims are stored in a portable form by the Subject or entity appointed by the Subject, retrieval by the owner of the data is trivial. Thus a Subject may have access to their data anytime in a similar fashion as they have access to any confidential digital documents. E.g storage on a mobile device may be retrieved with a thumb print access.

Verification of claims on the other hand is less trivial as the nature of the Verification algorithm may expose sensitive parts of the claims. In such a situation, the Signature Verification Paradox and Section 5.2.3 must be considered.

5.7.2 What are the minimum security requirements needed to enforce secure granular data management on the blockchain?

Granular data management here refers to data management process that exposes data to a minimum extent for a minimum amount of time only to desired entities. This is in fact a privacy constraint, and a security (confidentiality) constraint.

To achieve granularity, we propose **atomic signing** of claims. This means constructing a Merkle tree of hashes of all individual (atomic) claims (c_i) and then signing the Merkle root hash. With this technique, a person may for example expose only one claim (e.g their

address) with the claim hash and the Merkle proof without exposing other claims during verification. This is the direct opposite of bulk signing of claims (c[]), where a Subject has to expose all claims during Verification (no matter the Verification method).

With the granularity problem solved with atomic signing, what are the minimum requirement for security of each atomic claim. In this paper we express these requirements as a security game involving the Transformation Oracle and the Verification Oracle as shown in Section 5.3

5.7.3 To what extent is the chosen method in 1 above scalable?

The scalability of the method proposed by this paper can be viewed in two ways: claim issuing and claim verification. In order to accurately, calculate the degree of scalability, it is important to determine the minimum number of transactions or time taken to issue and verify a claim.

- **Claim Issuing:** Claim issuing involves two processes. Claim verification (Generate algorithm) and claim signing / attestation (Transformation algorithm). These two processes are efficient when performed in a setup phase off-chain. We argue that since claims may be generated and distributed in a peer to peer fashion, they are highly scalable. The only bottlenecks occur when a single Authentication agent (blockchain gateway) is performing multiple claim issues for a large number of clients. In this case, we argue that such large scale Identity providers already have large scale efficient networks for distribution. Even if they do NOT have efficient claim generation and distribution means, this bottleneck does not affect the entire Self-Sovereign Identity management system.

- **Claim Verification:** In decreasing order of speed of verification, claim verification can be performed in three ways: full disclosure, anonymized disclosure, and Zero Knowledge Disclosure. This ordering simultaneously doubles as increasing order of privacy. Therefore we argue that since verification is a trade-off between efficiency and privacy, scalability depends largely depends on the particular use case. For example, if privacy is not an issue, verification is a 3 step transaction:

  1. Initiating verification on blockchain
  2. Transmitting data (claims, signed claims) off chain
  3. Verifying signature of identity provider off-chain.

  Therefore scalability here depends largely on the speed of consensus algorithm of the blockchain used (transactions per second), since off chain transaction are very fast.

  On the other hand, if a Zero Know Knowledge verification is used, even though this process may be performed off-chain, the ZKP setup stage may take hours and the ZKP verification phase on-chain consumes about 20million gas on Ethereum.

Therefore, careful considerations about the factors affecting efficiency and scalability must be made when constructing the Verification algorithm or oracle.

5.7.4 Are there any other use cases for such a solution in 1 above?

At its height, a fully fledged self-sovereign Identity Management system is capable of generating trust almost instantaneously between strangers (2 or more entities with no prior knowledge of one another). At the time of writing, one pressing use case is in the drastic reduction of cost and delay in Know-Your-Customer(KYC) processes of banks.

However, this research found that there are many more use cases for this solution. A few of these are as follows:

- **Web applications that require strong authentication:** Websites and web application that require the user to prove any claims while simultaneously maintaining the user's privacy are one of these use cases. For e.g we found that in the UK, a verification tool has been introduced to regulate access to websites with adult content by proving one's age. With this use case, such verification proofs can be performed efficiently and pseudonymously while simultaneously preserving the Subject's privacy. [Bar]

- **Creating group claims:** If any group of people wish to create a digital "identity card" for the group members, one simple way to achieve this is by peer-to-peer signing of their claims. In this setup, each user is simultaneously a Subject as well as a claim issuer and verifier. For scalability, a threshold may be set on the number of verifications need to access a service restricted to the group members.

- **Internet Banks:** Internet banks may also use a blockchain identity for their on boarding services and services that require strong identification.

- **Real life services requiring strong authentication:** There are real life services like online car renting that require strong authentication on the service provider's side. However, on the user's side there is sometimes a requirement for privacy. In such a pseudonymous identity like a Self-sovereign blockchain Identity may be used. In summary, the service provider may not know the Subject accessing the service but is sure that if the Subject acts maliciously, someone (their Identity provider) can identify them.

5.7.5 How do we achieve compliance in use cases that require strong authentication (Example Governmental level, Banking)?

In high value use cases such as the banking sector, governmental oversight is required. For example in Europe KYC processes are regulated by regulations like the 4th and 5th Anti Money Laundering directive (AMLD). [PtC] Online services as of the time of writing are required to be GDPR compliant. [Conb]

Some of these regulations sometimes conflict with one another (e.g AMLD and GDPR). In terms of compliance to governmental oversight the current solution ensures that all transactions are logged and publicly available on the blockchain for compliance agencies. If there is a need for further information, the necessary legal procedures may be triggered with the Identity Provider in question. As an addition, the underlying blockchain used may also play a crucial role in compliance, thus, it may provide meta-data to governmental agencies. As to whether or not the blockchain provided meta-data is a breach of user privacy is out of the scope of this paper. In terms of GDPR, the stakeholder definitions and operations strongly align the clauses stated in the GDPR documentation, thus, a self-sovereign identity may be made fully GDPR compliant with little effort.

Chapter 6

Conclusion

This research defines and explores the problems of Identity Management Systems and outlines, with Cryptographic proofs, one way of achieving a self-sovereign identity with blockchain technology. The main problem was the incentive misalignment between Subject, Authentication agent, and Authorization agent caused by conflicting interests and responsibilities. For e.g. an Authentication agent storing the claims of a Subject (without providing oversight to the Subject) is a clear conflict of interest that provides a perverse incentive. The consequence of these problems are type 1, type 2 and type 3 attacks as shown in Section 2.1.3

Our solution begins by atomically defining the building blocks of an Identity Management system (claims, trust, and digital identities) and gradually builds it up into a full blown self-sovereign identity management system consisting of polynomial time algorithms (G,T, and V). The supporting cryptographic proofs for achieving security and privacy are expressed as a security game for the Transformation oracle (T) and Verification oracle (V).

The self-sovereign identity management system is then categorized into blockchain gateway and blockchain framework. These provide a clear and concise way of achieving **separation of powers** between Subject, Authentication agent, and Authorization agent. Our research also fills the research gap involving the technical details required to achieve GDPR compliance as well as meeting other regulations. In summary, this research establishes a secure and privacy friendly middle ground between the blockchain and the mundane world (off-chain) using a hybrid solution that comprises of a blockchain gateway and a blockchain framework.

It is our hope that this paper will serve as a guideline or whitepaper, providing a wide discretion for any designer of a secure Self-Sovereign Identity. The details of the Generation, Transformation, and Verification algorithms are up to the designer, however, they should be correlated with the security proofs and efficiency (time and cost).

6.1 Comparison with literature

During the research, a lot of literature was reviewed. While there was a strong correlation or alignment in most parts of the literature body, other parts of the literature body were in complete disagreement with our findings.

- **Correlations:** The following parts of our results had a strong correlation with the corresponding literature:

  1. **Formal Definitions:** Our formal definitions of Subject, Authentication agent, Authorization agent, claims, and attributes were in strong alignment with the definitions provided by the GDPR documentation [Conb], W3C [SL], ERC735 [Voga], ERC725 [Vogb], actors involved in eIDAS [Dige] [Digd] [Digc], eIDAS attributes profile[Digb], and many more.

     It should be noted that deferent terminologies were used to represent the same idea. For example, in the GDPR documentation, Data subject is used instead of Subject. Another example is the eIDAS attributes profile that categorizes Subject by the type of person: natural person, legal person, and representative of legal person. A natural person has 8 attributes (4 mandatory attributes needed for cross-border authentication) and a legal person has 10 attributes (2 mandatory attributes needed for cross-border authentication within the EU)[Digb],.

  2. **Blockchain gateway and framework:** There was a strong correlation between claim generation, storage, issuing, and attestation of the ERC725 [Vogb] and ERC735 [Voga] design and our design of the blockchain gateway and framework.

     However a critical difference worth mentioning is that the verification mechanism proposed by the ERC design leaks information. Immediately a Subject verifies his / her claims with a service provider, that Subject immediately loses control and transparency of transactions involving their claims. This is because the design by default uses a full disclosure verification Section 5.2.3. No details were provided by the ERC design as to how to measure or achieve Zero Knowledge disclosure verification Section 5.2.3.

     Therefore, we summarize that according the formal definitions and security proofs provided by this paper in Section 5.3, the ERC identity scheme is EU-CCA secure but not semantically secure.

  3. **Principles of self-sovereign identity:** The design of our solution fully satisfies principles 1 to 7 of the self-sovereign Identity principles stated in Section 2.2.7. Principles 8 and 9 depend on the details of the Verification algorithm chosen. We argue that if a Zero-knowledge disclosure (as described in Section 5.2.3) is used in verification, our design will also satisfy principle 8 and

9. To fully satisfy principle 10, some non-technical (example: legal) measures have to be taken.

- **Disagreements:** One particular area of our research was either in strong or weak disagreement with the body of literature. The following explains the disagreements and why our solution is still secure:

  1. **Trust:** One of the core primitives of our design of the blockchain gateway and framework was trust. In all the literature we researched, there was no formal definition for trust in relation to digital identities. Therefore, all prior work was done with a somewhat loose definition of trust which is equivalent to blind reliance on another entity.

     Carl Ellison in his paper, "Establishing Identities without certification Authorities" [E$^+$96], essentially establishes trust by sharing an actual secret between two participants each with a public-private key pair. Indeed, this aligns with our Lemma 5.1.2, however, it requires that the two parties share prior knowledge of each other. This prior knowledge is used as the secret. This is strictly not trust generation but rather an efficient way of authenticating the holder of a claim (public key). Trust generation as we define it requires zero prior knowledge between the parties involved.

     Another paper titled "Why trust isn't transitive?" [CH96] outlines some key reasons why trust is NOT transitive and why trust transitivity is a dangerous security assumption to make. Based on the definition of trust given in that paper, it is a perfectly reasonable conclusion.

     However, we argue that these two papers ([E$^+$96] and [CH96]) refer to another definition of trust, which either requires prior mutual knowledge between the parties in question or is blind and without time bounds. For this reason, we provided a formal definition of trust in Section 5.1.4 which is not necessarily blind and has a definite time bound in accordance with the explanation of Lemma 5.1.2.

     In summary, by our definition of Trust and the supporting lemmas, we argue that Trust as defined in Section 5.1.4 is indeed transitive as long as the claims shared are still valid under a particular verification algorithm trusted by the Service provider. This trust may securely act as a foundation for the self-sovereign Identity management system design proposed.

6.2 Future Development

Traditionally, it is common practice to write a will when one dies. This is done to ensure that the allocation of one's resources is consistent with the owner's wishes. Suppose Bob needs to allocate some ethers to his unborn child, how can this be achieved? One solution, of course, is to deposit the ethers in a smart contract that releases the funds upon a successful paternity test. But how do you authenticate the delivery of an asset to an identity that has not yet been created? And how do you trust that the oracle providing the paternity test results is authentic?

At the time of writing, it is clear that this is the future of this research: A way to achieve secure and authenticated data exchange between a smart contract and an off-chain oracle.

The solution to this could also be used to solve the key management problem if a user's keys are managed entirely by a smart contract. However, for this solution to work, the user needs to somehow communicate confidentially with the smart contract.

One final direction for future development of this research is the development of a more efficient Zero-Knowledge proof protocol for Zero-knowledge disclosure verification as proposed in Section 5.2.3.

Bibliography

[ALC+98] Hal Abelson, Lawrence Lessig, Paul Covell, Steve Gordon, Alex Hochberger, James Kovacs, et al. Digital identity in cyberspace. *White paper submitted for*, 6, 1998.

[All] Christopher Allen. The path to self-sovereign identity.

[AR97] Alfarez Abdul-Rahman. The pgp trust model. In *EDI-Forum: the Journal of Electronic Commerce*, volume 10, pages 27–31, 1997.

[B+02] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[B+14] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[Bar] Sabrina Barr. Porn website age verification tool officially announced within uk.

[BBB+] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more.

[BBHRSG17] Jorge Bernal Bernabe, Jose L Hernandez-Ramos, and Antonio F Skarmeta Gomez. Holistic privacy-preserving identity management system for the internet of things. *Mobile Information Systems*, 2017, 2017.

[BDP04] William E Burr, Donna F Dodson, and William T Polk. *Electronic authentication guideline*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2004.

[BDPVA09] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3(30), 2009.

[Bir] David Birch. How to use identity & the blockchain.

[BP04] Ben Bauer and Andrew S Patrick. A human factors extension to the seven-layer osi reference model. *Retrieved January*, 6:2004, 2004.

[But16] Vitalik Buterin. Chain interoperability, 2016.

[CH96] Bruce Christianson and William S Harbison. Why isn't trust transitive? In *International workshop on security protocols*, pages 171–176. Springer, 1996.

[Cona] ConsenSys. Are you really using sha-3 or old code?

[Conb] Intersoft Consulting. General data protection regulation.

[Dam] Micah Dameron. Rediscovering ethereum: The beige paper backstory.

[DB18] M Divya and Nagaveni B Biradar. Iota-next generation block chain. *International Journal Of Engineering And Computer Science*, 7(04):23823–23826, 2018.

[Diga] Digiconomist. Bitcoin sustainability report february 2018.

[Digb] CEF Digital. eidas saml attributes profile v1.1.2.

[Digc] CEF Digital. What are the benefits of eid?

[Digd] CEF Digital. What is eid?

[Dige] CEF Digital. Who is involved in eidas?

[DMH17] Vikram Dhillon, David Metcalf, and Max Hooper. Unpacking ethereum. In *Blockchain Enabled Applications*, pages 25–45. Springer, 2017.

[DT] Sebastian Zickau Dirk Thatmann. Identitychain. `http://www.snet.tu-berlin.de/menue/projects/identitychain/`.

[DW13] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.

[E+96] Carl Ellison et al. Establishing identity without certification authorities. In *USENIX Security Symposium*, pages 67–76, 1996.

[FH] Niels R Faber and Henk Hadders. Towards a blockchain enabled social contract for sustainability.

[FP12] Md Sadek Ferdous and Ron Poet. A comparative analysis of identity management systems. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 454–461. IEEE, 2012.

[Gar] Gartner. Planning guide for identity and access management. `https://www.gartner.com/binaries/content/assets/events/keywords/identity-access-management/iame11/2017-planning-guide-for-identity-and-access---13oct16.pdf`.

[Gar95] Simson Garfinkel. *PGP: pretty good privacy.* " O'Reilly Media, Inc.", 1995.

[Gru13] Sarah Gruber. Trust, identity and disclosure: Are bitcoin exchanges the next virtual havens for money laundering and tax evasion. *Quinnipiac L. Rev.*, 32:135, 2013.

[HB08] Ruth Halperin and James Backhouse. A roadmap for research on identity in the information society. *Identity in the information society*, 1(1):71–87, 2008.

[ibm] Ibm and securekey technologies to deliver blockchain-based digital identity network for consumers. `https://www-03.ibm.com/press/us/en/pressrelease/51841.wss?lnkcaenhpv18l1`.

[id2] Id202 alliance.

[JHF03] Ken Jordan, Jan Hauser, and Steven Foster. The augmented social network: Building identity and trust into the next-generation internet. *First Monday*, 8(8), 2003.

[KK03] M Frans Kaashoek and David R Karger. Koorde: A simple degree-optimal distributed hash table. In *International Workshop on Peer-to-Peer Systems*, pages 98–107. Springer, 2003.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[KRGP18] Sabine Kolvenbach, Rudolf Ruland, Wolfgang Garther, and Wolfgang Prinz. Blockchain 4 education. In *Proceedings of 16th European Conference on Computer-Supported Cooperative Work-Panels, Posters and Demos*. European Society for Socially Embedded Technologies (EUSSET), 2018.

[KRvW] Tommy Koens, Coen Ramaekers, and Cees van Wijk. Efficient zero-knowledge range proofs in ethereum.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[Med] Medici. 21 companies leveraging blockchain for identity management and authentication.

[MW] James A. Martin and John K. Waters. What is identity management?

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[Ped13] Mari Pedak. Estonian experience. 2013.

[Pre] Alex Preukschat. Self sovereign identity - a guide to privacy for your digital identity with blockchain.

[PtC] European Parliament and the Council. Directive (eu) 2015/849.

[Rab17] Kefa Rabah. Digital cryptoeconomics powered by digital cryptocurrency. *Mara Research Journal of Computer Science & Security-ISSN 2518-8453*, 1(1):107–131, 2017.

[RS13] Steven B Roosa and Stephen Schultze. Trust darknet: Control and compromise in the internet's certificate authority model. *IEEE Internet Computing*, 17(3):18–25, 2013.

[S+03] Nigel Paul Smart et al. *Cryptography: an introduction*, volume 3. McGraw-Hill New York, 2003.

[SdSSW95] Patrick Sénac, Pierre de Saqui-Sannes, and Roberto Willrich. Hierarchical time stream petri net: A model for hypermedia systems. In *International Conference on Application and Theory of Petri Nets*, pages 451–470. Springer, 1995.

[SL] Nate Otto Sunny Lee. W3c standards compare and contrast. `https://w3c.github.io/vc-use-cases/`.

[SPM+11] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. What makes users refuse web single sign-on?: an empirical investigation of openid. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 4. ACM, 2011.

[Sto] Cossack Labs Dev Stories.

[Voga] Fabian Vogelsteller. Claim holder. `https://github.com/ethereum/EIPs/issues/735`.

[Vogb] Fabian Vogelsteller. Ethereum request for comments identity mgt. `https://github.com/ethereum/EIPs/issues/725`.

[Wal16] MGCSA Walport. Distributed ledger technology: Beyond blockchain. *UK Government Office for Science*, 2016.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[XLD05] Di Xiao, Xiaofeng Liao, and Shaojiang Deng. One-way hash function construction based on the chaotic map with changeable-parameter. *Chaos, Solitons & Fractals*, 24(1):65–71, 2005.

[Yoh] Binu Yohannan. Blockchain - a blessing for kyc and aml. https://www.intellectdesign.com/insight/blockchain-a-blessing-for-kyc-aml.html.

[ZXDW16] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *Work Pap.–2016*, 2016.