UNSW
AUSTRALIA

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

Decentralized Data Sharing in Web 3.0

By

Suebtrakul Kongruangkit and Yu Xia

Thesis submitted as a requirement for the degree of

Bachelor of Engineering in Computer Engineering

Submitted: December 2019
Supervisor: Dr. Helen Paik and Dr. Xiwei Xu
Student ID: z5176891 and z5212108
Topic ID: XX00

Group Member Contributions

For this thesis, Suebtrakul and Yu have been working together since the beginning of this term. We allocated weekly meeting, shared resources, and discussed our ideas such as designing a proposed architecture.

In this report, Suebtrakul worked on an acknowledgement, table of content, technical background knowledge, and bibliography, while Yu were in charge of an abstraction, introduction, general background knowledge, proposed architecture, and a project plan. All of them were double checked and finalized by both of us at the end.

Abstract

This thesis examines different methodologies and designs that aim to realize the idea of decentralized data sharing. They all share similar goals but tackle the problem from different perspectives. Among the available approaches, this report will mainly focus on the following two: Social Linked Data (Solid) and Blockchain technology. This report will analyze the similarities and differences between them and propose a design a new system that leverages the techniques from the two communities. A case of application will also be proposed at the end to demonstrate the underlying technical solutions and benefits of this combined system.

Acknowledgements

The work of this thesis has been inspired by Dr. Helen Paik and Dr. Sherry Xu who encourage both of us to explore the scope of this project leading to an enhancement in our research skills and understanding towards a concept decentralized platform. Further support and encouragement have been given by Sin Kuang Lo and Su Yen Chia who provide us the recommendation, advice, and information about blockchain technology. At last, we would like to thank you our families and friends who always support us.

Abbreviations

DApps	Decentralized Applications
WWW	World Wide Web
DOM	Document Object Model
Solid	Social Linked Data
WAC	Web Access Control
XACML	eXtensible Access Control Markup Language
PEP	Policy Enforcement Point
PAP	Policy Administration Point
AMs	Attributes Managers
PIPs	Policy Information Points
PDP	Policy Decision Point
CH	Context Handler

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The World Wide Web was originally designed as a decentralized system. However, over the last decade or so, the Web has evolved from the static "Web 1.0" to more interactive "Web 2.0", and it is now dominated by some big service and platform providers. Personal data is hosted by a few big service providers and the end-users lose control over their personal data. The concept of Web 3.0 is proposed to change the Web into a fully open and decentralized data ecosystem while recognizing the end-users as data owners with total self-governance over their data.

There already exist some foundations and organizations experimenting on different methodologies and trying to propose a solution to realize the idea of the idea of decentralized data sharing. Some of them are re-implementing the whole Internet protocols and others are building new layers on top of the existing Internet. They all share similar goals but offer different capabilities and characteristics. This report will broadly discuss the different approaches that are currently available, but mainly focus on two of them: Social Linked Data (Solid) and blockchain technologies. The ultimate goal of this thesis is to propose a methodical solution that combines the two approaches so that it can result in new and improved concepts for decentralized data sharing system.

In this report, we will first present some background on different technologies, followed by detailed examination on Solid and Blockchain. Design decisions and rough architecture will be discussed in the third part. After that, a use case application will be proposed to effectively demonstrate the underlying technical solutions. The project timeline and future plans for thesis B and thesis C will be mentioned in the last part.

Chapter 2

Background

2.1 General Background

There exist three different phases during the evolution of the World Wide Web (WWW). In the first stage, namely Web 1.0, there were only a few content creators and websites were mainly consisted of static web pages without too much input from end users. Majority of the users were simply acting as content consumers (Graham Cormode, 2008). This stage is commonly known as the readable phase of the WWW with flat data. In late 2004, the world "Web 2.0" was introduced the first time to denote the second phase of the WWW. At that time, new technologies and concepts such as AJAX and Document Object Model (DOM) were invented to help users better interact with web applications. Websites tend to incorporate strong social components, which involves user profiles, comments and friend links (Graham Cormode, 2008). People were encouraged to upload their own personal-generated data such as videos or blog posts along with tags and comments. Compared with Web 1.0, Web 2.0 brings more interactions between users and websites, and users are treated as first class entities rather than passive content consumers.

One of the important features of Web 2.0 is that user data is stored in the backend databases of applications, which is hosted and controlled by a few big service providers like Google Cloud or AWS. This centralized fashion could potentially raise many problems. One of them is that users lose control of their personal data. At the moment when they upload their pictures and posts on social media, they cannot make decisions of who can access or share this piece of data. Users have to sacrifice data privacy in order to use applications. Although many web applications would normally have terms and agreements, it is still considered as a high risk to share personal information from users' perspective. Another issue is that those centralized web

servers and databases are not completely safe. Data leakage and database hacking could lead to large scale security accidents. For example, one of the largest credit agencies in the U.S, Equifax, suffered a serious data breach that affect 143 million consumers in 2017 (Newcomb, 2017).

Due to all the potential risks that Web 2.0 might have, the concept of Web 3.0 was introduced to address those problems during the past decade. There is no concrete definition of Web 3.0 yet, different people would have different opinions about the future web. According to Eric Schmidt, ex Google CEO, Web 3.0 will be "applications that are pieced together, with the characteristics that the apps are relatively small, the data is in the cloud, the apps can run on any device, the apps are very fast and very customizable, and are distributed virally" (Macmanus, 2007). The core idea is that data is stored in the cloud or controlled by the owner, and data should be decoupled from applications in order to make applications more flexible, extensible and decentralized. There are already many organizations and research teams aiming to approach this goal, with either similar or completely different ways. In the next section of the report, different approaches will be examined and compared with each other.

2.2 Technical Background

There are several technologies that aim to create infrastructures for decentralized data sharing platforms. However, only a few of them has been adopted because of the discussions in technical details about this area are still limited to researchers and practitioners (Andrei Vlad Sambra, 2016). This section presents the technologies that trying to provide an infrastructure for decentralized platform including Ethereum, Elastos, and Solid. Then we add more elaborations and discussions between these technologies. Finally, we present our problem statement for the thesis.

2.2.1 Ethereum

Ethereum is an another technology trying to approach the idea of decentralized platform and Web 3.0 by providing an infrastructure for running decentralized applications (Dapps) worldwide (**Buterin**). Decentralized applications are a piece of software that could communicate and interact with blockchain, in order to maintain the application

itself. This process can be done by using a programmable blockchain network that could support smart contracts, which allows people to trust their codes. To elaborate in more detail, it provides blockchain network with a built-in Turing-complete programming language, known as Solidity, that enables people to create a programming logic as a smart contract and deploy it on the blockchain (Wood, 2017).

2.2.2 Elastos

Elastos is considered as one of the technologies that almost achieve the concept of Web3. It has an almost successfully implemented decentralized ecosystem by running a virtual machine on top of existing internet protocol. According to the original philosophy behind Elastos's design, it aims to recreate a new WWW (World Wide Web) system which is influenced by a concept of blockchain. With an immutable feature of the data stored on blockchain, it is significantly easier to maintain and control digital assets stored in this network because all the assets are identifiable and traceable foundation (2018). In order to achieve this goal, Elastos has to be a platform for decentralized applications (DApps), that have no centralized server but runs on a P2P network instead.

Elastos adopts the concept of Bitcoin blockchain and Ethereum blockchain to create a vertical implementation of the whole architecture. Bitcoin blockchain provides a trust in data by having decentralized and immutable ledger, while Ethereum provides a trust in code by having a smart contract supported by programmable blockchain. The smart contracts enable users to have an automatically executed logic without a need of intermediary or centralized server to run it. Without intermediary, many problems in the system are solved such as data breaches or frauds. However, Ethereum infrastructure still has some limitations (foundation, 2018).

- **Speed and Storage** - Storage of the blockchain is very limited and the speed is very low since Ethereum blockchain creates a new block every 30 seconds. If users do not pay enough amount of gas, they probably have to wait for their data to be stored in the high congestion period.

- **Cost** – User has to pay a fee for every single execution of the smart contract in Ethereum infrastructure.
- **Flexibility** - Ethereum Virtual Machine (EVM) that is in charge of executing smart contract is coupling with the blockchain and make them inseparable. If there is a change on one side, it impacts the other.
- **Security** - even the executions of the smart contracts are on the chain, risks of being attacked still exist because its application is still on the internet.

It is obvious that not all the features on the application can be done through smart contracts due to the above problems. That is why most of the existing blockchain-based applications are financial related because other areas might not be convenient because no matter performances of machines are, it cannot speed up the Ethereum's computation (foundation, 2018). To summarize, existing blockchains are created for consensus-based record keeping which have an absence of computation speed. Moreover, the blockchain itself are not designed to store data but to record transactions, which is not possible to have enough space in storing digital assets. This results in making Ethereum not fully compatible to run decentralized applications.

As Elastos is a virtual machine running on top of the existing internet protocol, resulting in being completely separated from it. The system consists of four main pillars:

1. **Elastos Blockchain** - this is a main blockchain that allows services to provide a trust on the system.
2. **Elastos Runtime** - a lightweight operating system that requires an internet to run virtual machine. This could prevent the services from accessing the internet directly.
3. **Elastos Carrier** - it is a P2P-based decentralized platform connecting each node together, which acts as a network layer of the system instead of internet protocol (IP).
4. **Elastos Software Development Kit (SDK)** - SDK is used to access Elastos Carrier services on the smart web.

Due to these four main pillars, Elastos becomes a platform that runs on top of its own blockchain and connects through Elastos carrier protocol. All resources stored inside this

platform are considered as digital assets and all of them are traceable, which results in promoting the property rights of digital contents. Additionally, Elastos has predefined sidechains. It not only solves the scalability and speed problems in Ethereum, but also makes the main public chain clean and simple. Apart from that, Elastos runtime, that separates applications from the internet network by running everything on virtual machine, brings security and prevents leaked digital contents.

Bitcoin	Trustworthy Ledger
Ethereum	Trustworthy Ledger + Smart Contracts
Elastos	Trustworthy Ledger + Smart Contracts + Monetizable Dapps and Digital Assets

Table 1. quick summary of Bitcoin, Ethereum, and Elastos (foundation, 2018**)**

With all these features, Elastos provides a decentralized infrastructure having high scalability, security and trustworthy where all the digital assets and user actions are traceable. However, there is one important downside. As, it has achieved many features towards Web3.0 concept, especially being completely separated from current internet services. This platform will not be able to be used with all services in the existing internet.

2.2.3 Solid

Solid (Social Linked Data) is a platform that has been trying to approach the idea of decentralized web technology, especially in a data-sharing system perspective, by decoupling user data storage from applications. It aims to provide independence of data and mechanism of data management. This decentralized platform is implemented mainly for social applications. In this platform, users store their data inside their online storage space called *personal online datastore* (pod). It is an accessible storage service that is deployed either on individual servers or public servers maintained by pod providers (such as Google Drive). Each user could own more than one pod, choose between different pod providers, and be able to switch between them since various pod providers can result in different degrees of privacy, reliability, or legal protection(Andrei Vlad Sambra, 2016). Figure 1 shows the differences between centralized and decentralized applications in the data storage perspective.

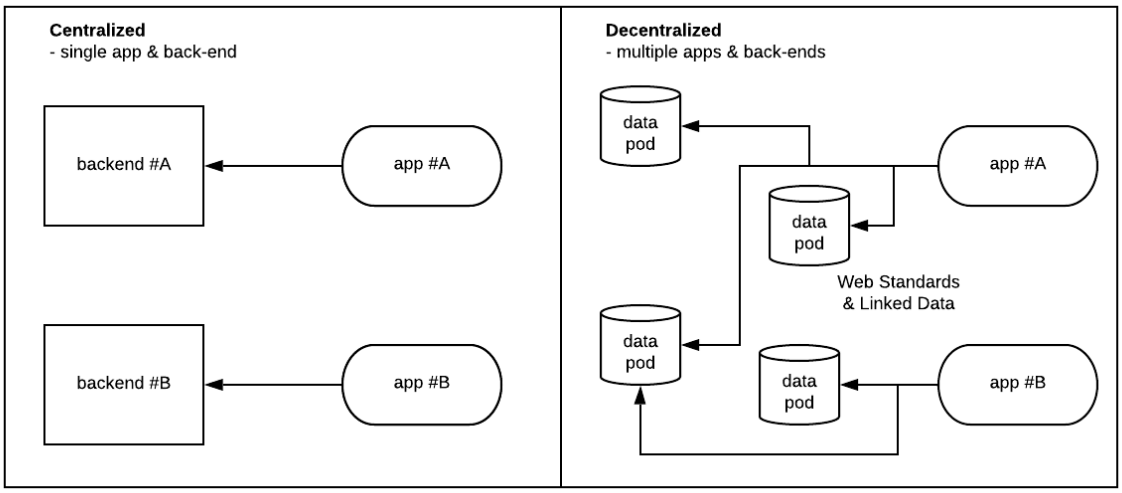

Figure 1. Differences between centralized and decentralized apps.

In the technological implementation perspective, solid is a decentralized framework that sits on the top of the existing internet infrastructure. It consists of many web standards, for example, WebID for identity system and user authentication, WAC for access control, Linked Data Platform for data manipulation, and SPARQL for complex data retrieval. All of them are integrated in a systematic manner, which results in offering standards-based encouragement and more convenient to be adopted by developers.

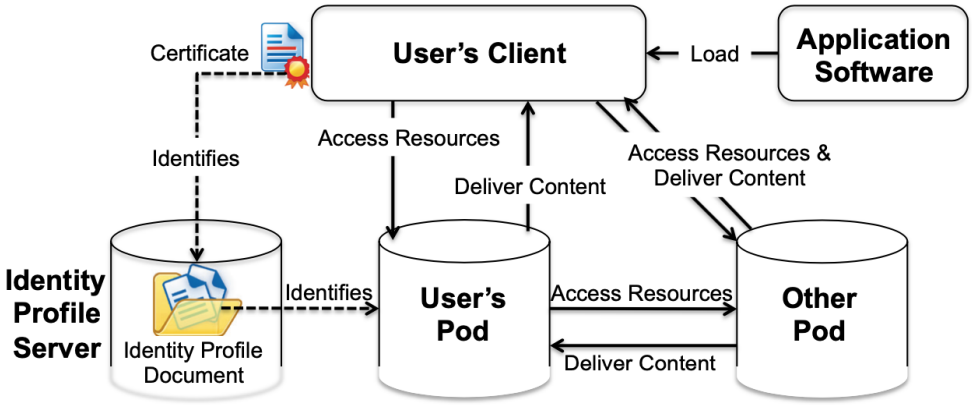

Figure 2. Solid Architecture

The figure above shows a client-side architecture of solid. Firstly, user downloads a solid application from its provider. Then, user's identity profile is used in order to access their pods. Some authentications are performed when it is necessary.

For the access control system in solid architecture, it adopts the idea of WebAccessControl (WAC), a decentralized cross-domain access control system, to describe the access control. There are different types of access control mode in WAC standard, including Read, Write, Control, and Append (community, 2016). With this access control standard,

it makes the system, which is LDP based, more convenient since not only all the resources are specified by URL, but also user identification is identified by WebID (URIs) as well (community, 2016).

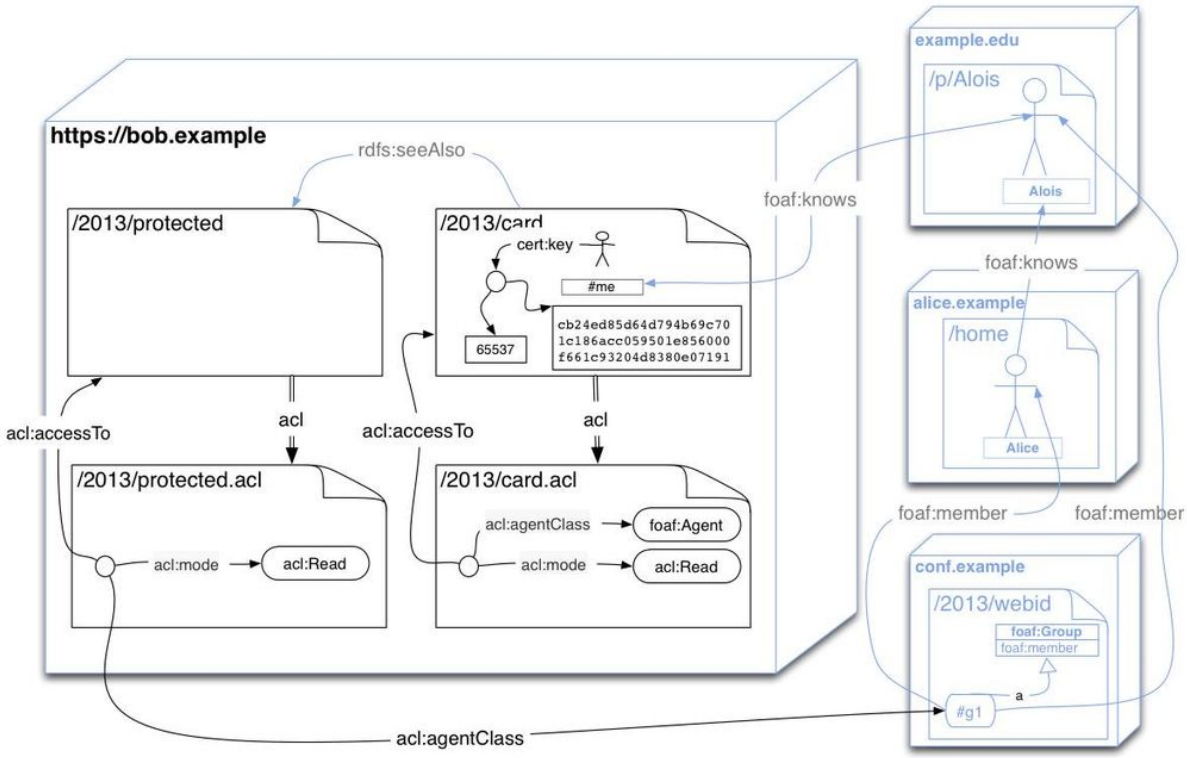

Figure 3. WebAccessControl (WAC) Standard

In the systems that use Web Access Control, they concern two factors, including agents and what type of access they have. The agents can be either a single user or a group of them, which are identified by WebID URIs as illustrated in the figure 1.

```
# Contents of https://alice.databox.me/docs/shared-file1.acl
@prefix  acl:  <http://www.w3.org/ns/auth/acl#>.

# Individual authorization - Alice has Read/Write/Control access
<#authorization1>
    a             acl:Authorization;
    acl:accessTo  <https://alice.example.com/docs/shared-file1>;
    acl:mode      acl:Read,
                  acl:Write,
                  acl:Control;
    acl:agent     <https://alice.example.com/profile/card#me>.

# Group authorization, giving Read/Write access to two groups, which are
# specified in the 'work-groups' document.
<#authorization2>
    a               acl:Authorization;
    acl:accessTo    <https://alice.example.com/docs/shared-file1>;
    acl:mode        acl:Read,
                    acl:Write;
    acl:agentGroup  <https://alice.example.com/work-groups#Accounting>;
    acl:agentGroup  <https://alice.example.com/work-groups#Management>.
```

Figure 4. Example ACL resource for individual and group authorization

The authorization can be specified either for a single resource, or a group of them according. For a group of resources, it is suitable to have resources being able to inherit permission from their folder/container because of the nature of web documents being hierarchical structure (community, 2016).

As illustrated in Figure 2, the identity profile server is the part that maintains all the access control of each pod, which can be either on a personal server or pod provider that runs the pod. Although solid provides a robust and decentralized data sharing storage system, its access control is still more likely to be centralized. According to the solid architecture that allows everyone can run a solid server themselves or for others, this results in solid being more like a federated system. However, few users will run their server while the most will use service providers to host their pods, which means the provider will own access control of these pods. We can consider this point as a centralized access control system that users have to rely on one third-party company in maintaining the access control system.

2.2.4 Blockchain-Based Access Control with XACML standard

According to the existing solid architecture, it does not adopt any idea of blockchain to provide decentralized features at all, which results in the remaining centralized access control system as service providers act as intermediaries maintaining access controls. As blockchain could replace a centralized server, there are some previous works (Damiano Di Francesco Maesa, 2019) trying to bring in a concept of blockchain and make use of a smart contract to the access control system.

(Damiano Di Francesco Maesa, 2019) proposed the idea of fully implementing blockchain-based access control with eXtensible Access Control Markup Language (XACML standard, defined by OASIS consortium (Standard, 2013)). XML-based language is explicitly designed for Attributed-Based Access Control (ABAC). It allows the user to write complex conditional policies corresponding to several attributes. For XACML architecture details, it is illustrated in Figure 5. There are several terminologies needed to be known.

PEP	**Policy Enforcement Point** pairs with resources and intercepts a request.
PAP	**Policy Administration Point** manages access control policies.
AMs	**Attributes Managers** manage attributes, allow to retrieve, and update them.
PIPs	**Policy Information Points** interacts with AMs
PDP	**Policy Decision Point** use policy, access request, and attribute values as inputs and then decide whether to provide permitted or denied access.
CH	**Context Handler** interacts with other components.

Table 2. XACML components

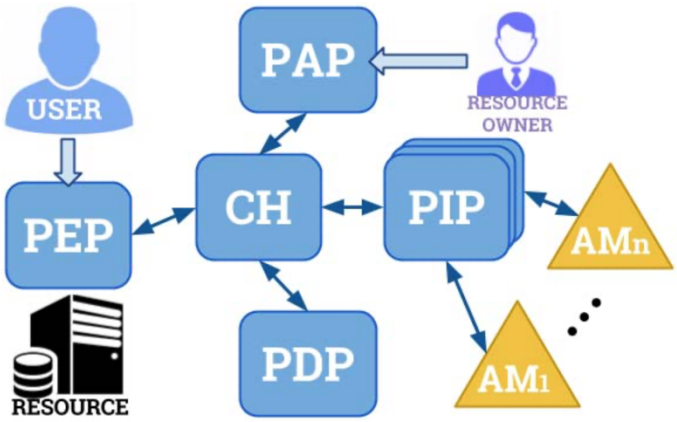

Figure 5. Simple XACML architecture

In the simple XACML system, once a user sends a request to access a resource, it is intercepted by PEP. The PEP converts the request and forwards it to CH. It extracts meta-data about the resource to retrieve attributes needed from PIP. Once all the attributes have been received, CH asks PDP to make a decision whether to grant or deny the access request.

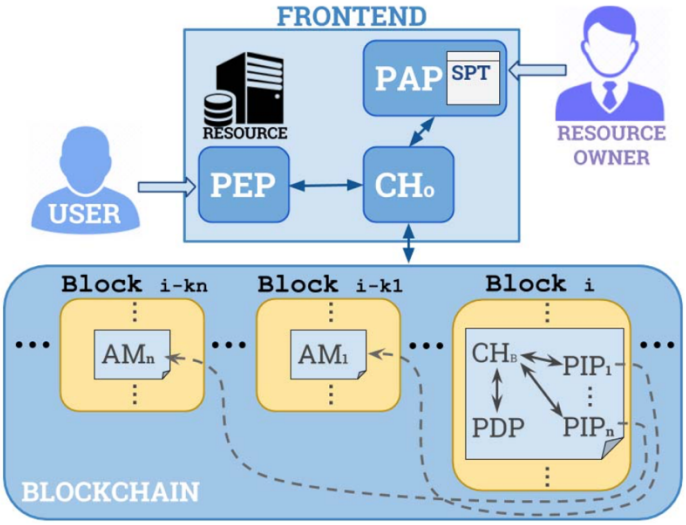

Figure 6. Architecture of the blockchain based Access Control Service.

For the proposed architecture, the figure 6 illustrates how blockchain replaces the XACML system. Access control system and policies are replaced with smart contract which is called *smart policy*. A transaction is created on the chain each time an access request is issued, and then PAP converts the logic for XACML policy to a smart policy. The smart policy contains all logics for executions, such as policy evaluation and attribute retrieval. In other words, PIP and PDP are done at once inside the smart policy. Once the

blockchain has stored the smart policy, its address is stored on the smart policy table (STP) by PAP.

Once there is an access request, PEP intercepts the request similarly to the simple XACML system. A request is forwarded to an off-chain context handler regarding a requested resource. It interacts with PAP and gets the smart policy address corresponding to the resource from STP. Then the smart policy is executed to perform the rest of the process including retrieving attributes and making decision of the request. In order to have an updated value of the attributes for PDP to make an access decision, smart AMs is created. It contains callable attributes retrieval function. This AMs are called by smart policy and all attributes needed are forwarded to PDP. Lastly, the smart policy also has a logic to disable itself, which can be called only from the owner because blockchain transaction is immutable, but can be marked as disable (Damiano Di Francesco Maesa, 2019). With this architecture, the whole access control can be moved into the blockchain.

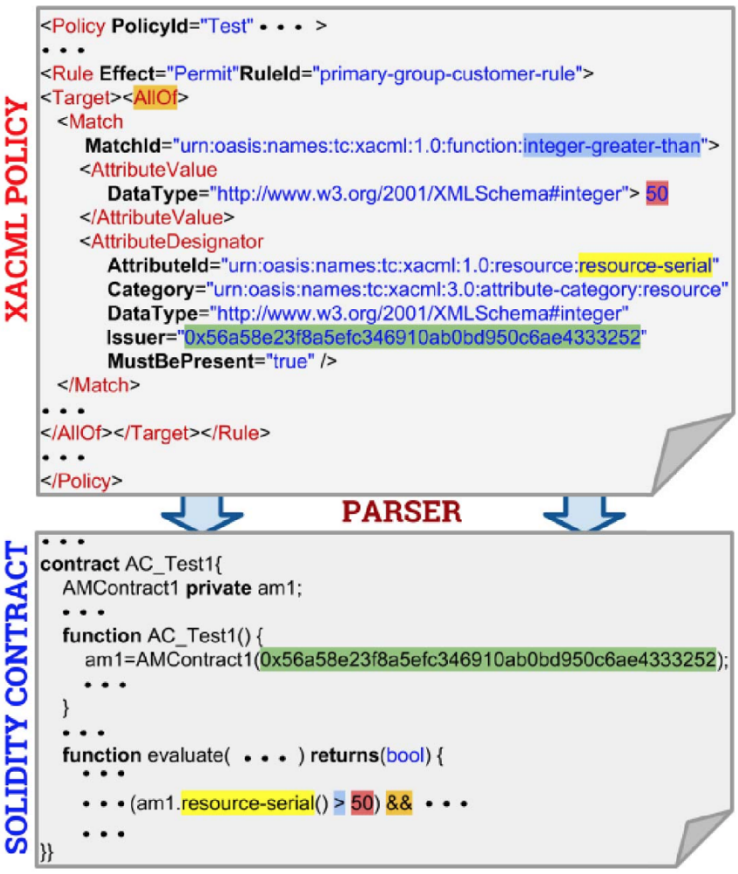

Figure 7. Simplified XACML to Solidity parser example

To discuss between WAC and XACML standards, XACML could provide a higher level

of complexity of access control policy as being attributed-based access control. However, the system that maintains it is more complicated, while WAC brings simplicity to the platform, especially when collaborating with Solid technology that has all the resources and entities are specified by URLs. We could adopt the idea of translating the XACML policy into a smart policy and store it into the blockchain by applying it with WAC.

2.3 Problem Statement

Due to the solutions provided by different technologies and communities, they approach the same problem from different angles. **Ethereum** comes with an infrastructure that allows developers to create and deploy their decentralized applications to the network. Despite this, some limitations are indicating that this platform is not practically compatible with running Dapps for several reasons, such as computational speeds, storage spaces, and cost constraints. **Elastos** solved this problem by designing a whole infrastructure with a vertical perspective. All the resources inside the network are identifiable and traceable. Moreover, all services inside this platform are decentralized with high scalability and efficiency. This is the platform that can be considered as a completed web3.0 platform. However, it aims to replace current internet platforms, not to solve it. Consequently, none of the existing services could connect with Elastos at all. **Solid** approach the Web3.0 specifically only in the decentralized data system. It aims to solve the existing centralized web platform by designing architecture with LDP, resulting in decentralized linked-data-based data storage, in spite of an access control system, which remains a nearly centralized system.

In this project, we focus on a decentralized data sharing system by adopting the solid infrastructure then designing an architecture that could bring blockchain and smart contracts in and solve the problem of a centralized access control system, while trying to keep existing access control system (WAC).

Chapter 3

User Case Scenario and System Design Overview

At this stage of the thesis, the aim of this report is not about proposing a unified system that complete combines all the good parts from Solid and blockchain technology. This report will only focus on a specific use case and will discuss how to address some common issues like identity management and resource access control within this specific use case scenario. But ideally, the architecture should be applied to more general cases. Further extensions and modifications will be made in thesis B and thesis C to achieve this ideal goal. The application that will be discussed is an assignment submission platform for university students.

In this assignment submission scenario, there exists some challenges that cannot be trivially solved by the traditional Web 2.0 technologies or the Web 3.0 platforms that discussed above but can be naturally addressed by some features of blockchain technology. Some of the challenges including:

1. Students cannot gain 100% access control of their own data, even the data belongs to themselves. Because of the university rules and regulations, student will have to give out some level of access control back to the university. For example, university requires to store student's assignment file for at least one year, which cannot be simply rejected by students.

2. Students cannot submit assignment once the due date has passed.
3. Students cannot share assignments with other students.

These problems are not very hard to address using traditional Web 2.0 technology. In fact, they can all be taken care of by some complex SQL statement or careful system design. But applying blockchain along with smart contract could be a more concise and elegant way to address this kind of access control problems. Below is the rough architecture of this assignment submission system.

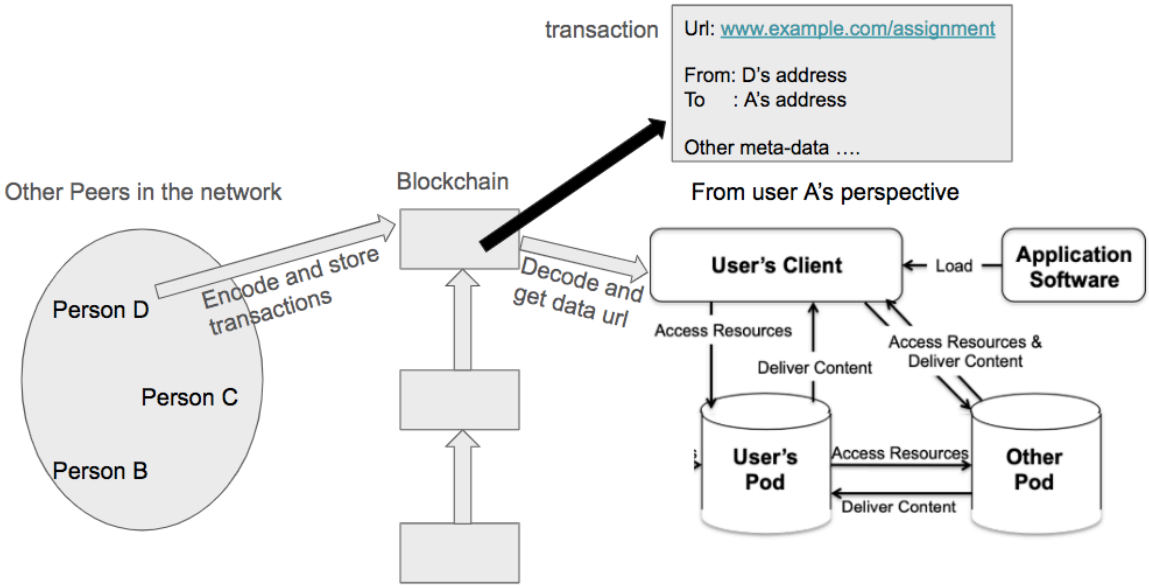

Figure 8. Proposed Architecture

The left half of the architecture comes from the original Solid design. Suppose user A opens a browser and loads an application, the frontend user interface of this application will show up on the screen. But at this point of time, data have not been filled in yet. Application would require to access user's pod to load some personal data, such as pictures or posts. Before this application can go into other people's pod and find what it needs, it must go through a pod server provider to check whether it has the right access level. In the above design, blockchain now replaces the original pod server. The core idea is to use an encryption algorithm to encrypt necessary information, which can only be decrypted by the people that have the right access. Below is a concrete example.

For instance, suppose user A wants to share a file with user B and user C. This encryption algorithm will take the keys from both user A, B and C as a source to encrypt some important

meta-data. And this algorithm has the property that the produced text can only be decrypted by one of the keys from the source. That is, the encrypted text can either be decrypted by user A's key, user B's key or user C's key, but not anyone else's key. Once the encrypted text is generated, it will be treated as a transaction and stored in blockchain. At that point, all the peers in the network will try to decrypt the transaction using their own key. Most of them will fail, because the text is not encrypted using their key, only A or B or C can decrypt it and access the actual content.

The meta-data would contain all the necessary information that describes what access level B and C will have, such as read access, write access or how long the access will last. The most important information is the link of data that A wants to share. In this way, once B or C decrypted the transaction, they can see the link and follow the link to retrieve actual content. Other peers cannot decrypt the transaction thus they do not have the data link that A wants to share.

There are different levels of access control, such as read, write, modify, append, etc. Writing, modifying or appending operations can be classified as one single category, because they all change the actual content. Thus, for the simplicity of this discussion, the next section will only focus on read and write access.

For giving read access of some data to other people, the whole process is relatively easy and straightforward, it is exactly the same as discussed above. Using that algorithm to encrypt information so that only those people who have right access can decrypt it. Below is a diagram that illustrates the process.

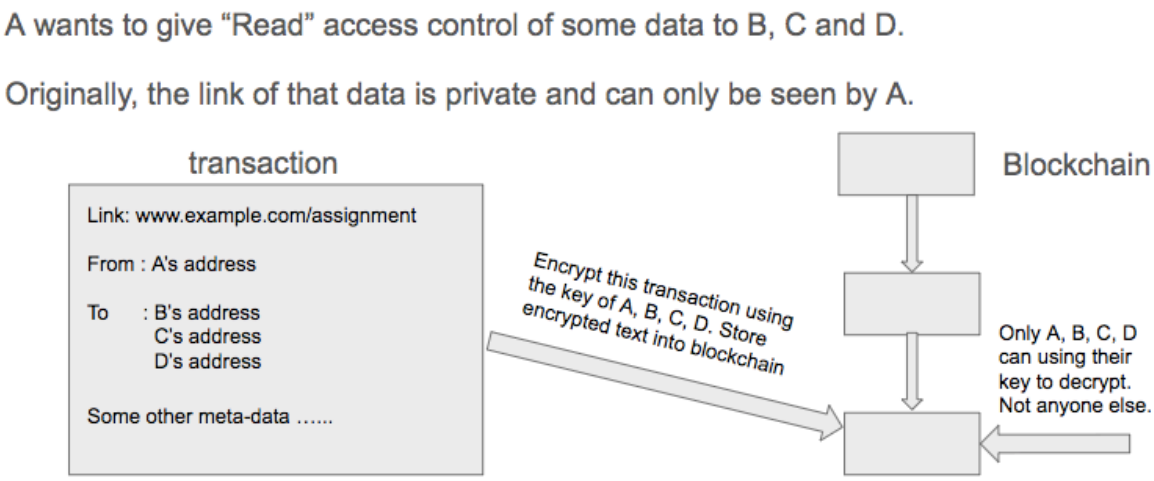

Figure 9. Process of Giving Read Access

Giving write access of a file to someone is a little different from giving read access. When multiple people trying to write some data to the same piece of data, that would sometimes cause data race. To address this issue, duplicate files will be created and assigned to each individual, so that they can apply write operation on their own copy. That is, when user A wants to give write access of some file to user B, take the keys of A and B as a source, feed to the encryption algorithm and then apply this algorithm to decrypt the original data link. The encrypted text would be a new link that denotes to a duplicate file that allow B to write on. Similarly, when user A wants to give write access to user C, apply the same methodology and take the keys of A and C to produce a duplicate file, which can only be accessed and written by C. In this way, each person would have their own isolated copy of file that allow them to write on without potential data race. The next step is the same as above, using the key of A, B, C and D to further encrypt the whole transaction and store in blockchain. The decryption process is slightly different as in giving read access, for each user of B, C or D, they now need to decrypt twice instead of once. After decrypt the whole transaction from blockchain, they will also try to decrypt each link and find the right one that belongs to themselves.

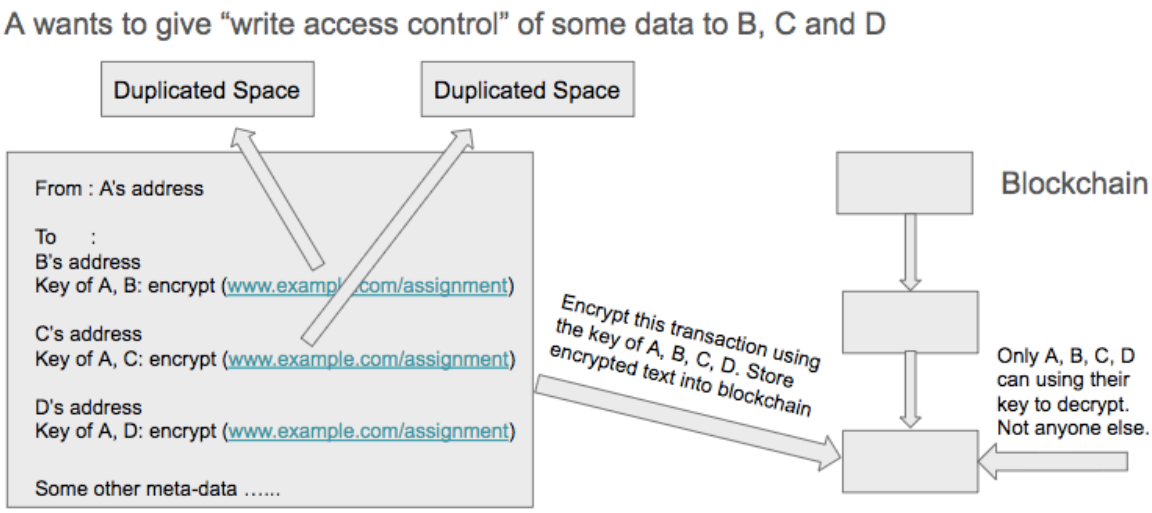

Figure 10. Process of Giving Write Access

One other scenario is when user A wants to give read access to some users but write access to some other users. In this case, the design would simply be a mix of two designs above.

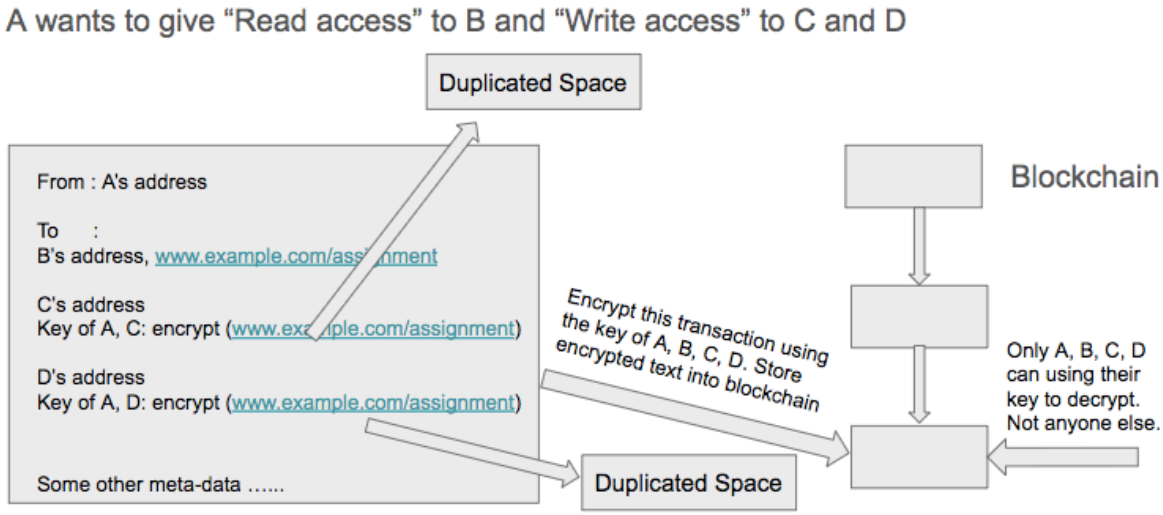

Figure 11. Process of Giving Read and Write Access

Merge all those small details together, below is the rough architecture for the whole assignment submission system.

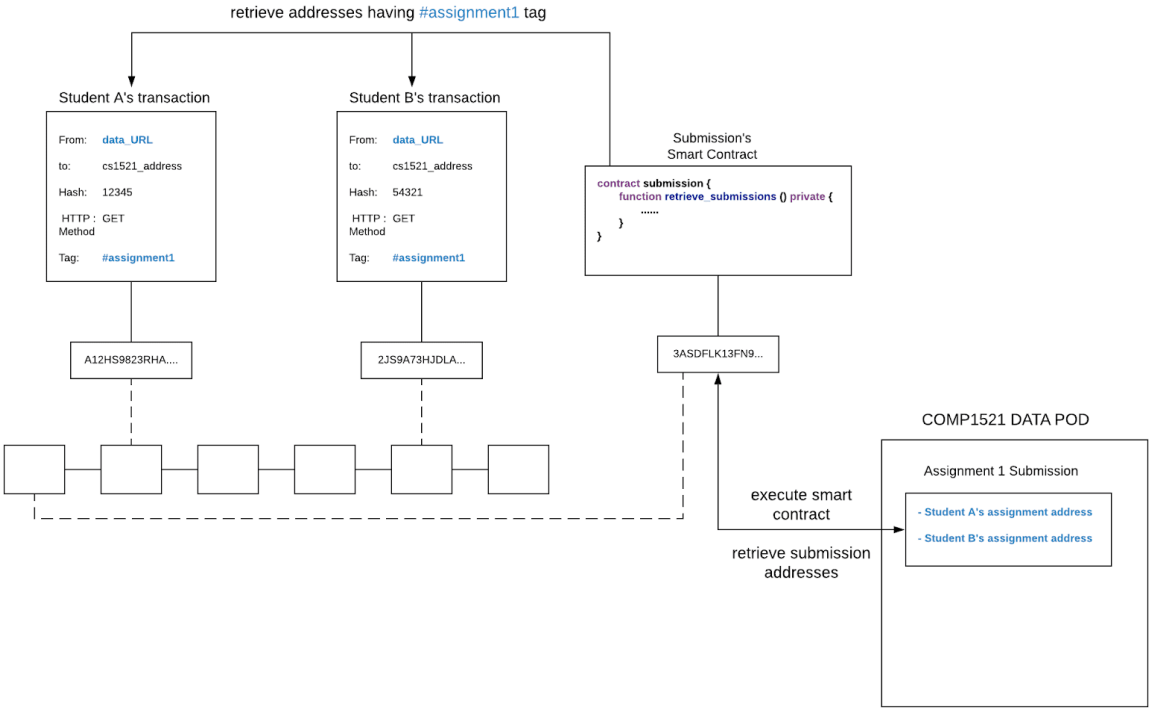

Figure 12. Using Smart Contracts in Assignment Submission

Each transaction would include the meta-data that discussed above. Once this transaction is created and appended to existing blockchain, every peer in the network will try to decrypt it using their own key. Due to the property of the encryption algorithm, some of them would succeed, but others would fail. Successfully decrypting the transaction means having the right access, and this peer is the right peer that the original person wants to share data with. Smart contract plays a role as bootstrapper and retrieve data from transactions and send to the right pod.

Chapter 4

Project Plan

So far, the work done in thesis A is the background research of different platforms. This report examined some existing platforms which trying to approach the idea of Web 3.0 and proposed a new architecture. The design is not completely finalized yet at this stage, it is only a rough prototype that could potentially work. Below is the timeline chart for future thesis research.

<table>
<tr><th></th><th>Pre-Preparation (Summer)</th><th>Thesis B</th><th>Thesis C</th></tr>
<tr><td>Week 1</td><td rowspan="3">1. Read and analysis a few existing Solid applications' source code.</td><td rowspan="2">1. Should have a fully functional application when Thesis B starts</td><td rowspan="2">1. Generalise the access control part</td></tr>
<tr><td>Week 2</td></tr>
<tr><td>Week 3</td><td rowspan="4">2. Refine current access control architecture</td><td rowspan="4">2. Extend the architecture to work with other applications, e.g. conditional / negotiable access control in doctor-patient relationship</td></tr>
<tr><td>Week 4</td><td rowspan="3">2. Explore APIs that Solid provides and write user stories for assignment submission application</td></tr>
<tr><td>Week 5</td></tr>
<tr><td>Week 6</td></tr>
<tr><td>Week 7</td><td rowspan="4">3. Build the assignment submission application entirely based on Solid</td><td rowspan="4">3. Refactor the codebase to adopt blockchain-based access control</td></tr>
<tr><td>Week 8</td><td rowspan="3">3. If time allows, build more applications on this platform, e.g. student resume generating functionality</td></tr>
<tr><td>Week 9</td></tr>
<tr><td>Week 10</td></tr>
</table>

Table 3. Project Timeline

The plan is to first build the assignment submission application entirely based on Solid platform, which could give us the chance to look inside the source code and gain a deeper understanding of how Solid access control part works. This should be done during the summer break. A concrete and fully workable application should be built before Thesis B

starts, and then further discussion and improvements can be made based on that. Thesis B will mainly consist of refactoring the codebase to adopt new access control models, which should be relatively easy to do if the application is built with good software development methodologies. In thesis C, the goal is to generalize the access control part. Ideally, it should not only limit to this specific assignment submission scenario but can be applied to other use cases as well.

Apart from the traditional access control mechanism, there exists some other interesting use cases in our real life. One good example occurs in the doctor-patient relationship. It would be nice if we can apply smart contract to deal with the negotiable or conditional access control mechanism, which could give patients more personal privacy and gain more control of their own data. This is the area that we will also explore in thesis C.

Bibliography

ANDREI VLAD SAMBRA, E. M., SANDRO HAWKE, MAGED ZEREBA, NICOLA GRECO, ABDURRAHMAN GHANEM, DMITRI ZAGIDULIN, ASHRAF ABOULNAGA, TIM BERNERS-LEE 2016. Solid: A Platform for Decentralized Social
Applications Based on Linked Data.

BUTERIN, V. Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform.

COMMUNITY, S. 2016. *Web Access Control (WAC)* [Online]. Available: http://solid.github.io/web-access-control-spec/ [Accessed].

DAMIANO DI FRANCESCO MAESA, P. M. 2019. Blockchain Based Access Control Services.

FOUNDATION, E. 2018. elastos white paper: Smart-web powered by blockchain.

GRAHAM CORMODE, B. K. 2008. Key differences between Web1.0 and Web2.0.

MACMANUS, R. 2007. *Eric Schmidt Defines Web 3.0* [Online]. Available: https://readwrite.com/2007/08/07/eric_schmidt_defines_web_30/ [Accessed].

NEWCOMB, A. 2017. Massive Equifax Data Breach Could Affect Half of the U.S. Population.

STANDARD, O. 2013. *eXtensible Access Control Markup Language (XACML) Version 3.0* [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html [Accessed].

WOOD, G. 2017. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER.