INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Data Protection with Ethereum Blockchain

DUARTE PAULO GONÇALVES CABRAL TELES
Outubro de 2018

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Data Protection with Ethereum Blockchain

Duarte Teles

Dissertation to obtain the Master's Degree in Informatics Engineering, specialization in Computer Systems

Supervisor: Isabel de Fátima Silva Azevedo

Porto, 14th October, 2018

Dedicatory

To my family and friends, whom without their support this document would have not been possible.

Abstract

Blockchain technology has been one of the most promising technologies of the past decade, with Ethereum and Bitcoin being the two most popular Blockchains today. Both do not provide data protection and privacy by default. The former allows for Decentralized Applications (DApps) to be built, with zero chance of downtime or censorship and is the main focus of this dissertation.

The European Union approved a law in 2016, the General Data Protection Regulation (GDPR), with severe penalties being enforced since May 25th, 2018. It is considered a massive step toward protecting user data. Not only does it affect companies with offices in the EU, but also organizations throughout the world that have users from EU territories. Further, it stipulates key obligations for organizations handling user data, in addition to introducing new rights to individuals, such as the right to erasure. This represents a major challenge towards achieving GDPR compliance in DApps, as Blockchains such as Ethereum, are immutable by design.

This dissertation's work attempts to comply with the GDPR and its conflicting right to erasure, by developing an Ethereum proof-of-concept DApp: DFiles, which also aims to provide some form of data privacy and protection. It also allows its users to upload encrypted files in addition to their download and decryption. It was developed using an Agile methodology in an iterative approach with mainly decentralized technologies, such as the Interplanetary File System (IPFS) and Ethereum smart contracts, with a centralized component for user authentication, while adhering to Blockchain Software Engineering. Due to the GDPR's complexity, some parts were selected, namely the rights to erasure, data portability, access and rectification.

DFiles GDPR compliance was then evaluated with a statistical analysis on user encrypted and unencrypted uploaded files in the DApp, with its elapsed upload times and Ethereum transaction costs measured for files separated into four categories: **small** (1KB-1MB), **medium** (1MB-20MB), **large** (20MB-200MB) and **extra-large** (200MB-2GB). However, due to hardware limitations, this statistical analysis could only be performed for files up to 14.2MB. It concluded that transaction costs for unencrypted files are slightly higher, although this increase is not significant. As for elapsed upload times, it found that the elapsed upload time in encrypted files was overall significantly higher. Data from files larger than 14.2MB was still recorded which determined that the last two elapsed upload times for unencrypted files up to 800MB, are less than the last two upload elapsed times for encrypted ones up to 14.2MB.

In conclusion, encrypting files to comply with the General Data Protection Regulation's right to erasure is a valuable option only for small to medium files up to 14.2MB. From there,

without considering hardware encryption limitations, upload times tend to grow exponentially. Ethereum and the IPFS must advance to allow better privacy techniques. Recently, there have been major new improvements to Ethereum and its smart contracts; the world of DApp development is always changing at a fast rate. In the future, Ethereum might evolve to a newer version which may bring new and enhanced privacy controls which may allow its complete GDPR compliance.

Keywords: Ethereum Blockchain, General European Data Protection Regulation, smart contracts, data privacy and protection, decentralized applications, Blockchain software engineering, Interplanetary File System

Resumo

A tecnologia de *Blockchain* tem sido uma das mais promisoras da última década, com *Ethereum* e *Bitcoin* como as duas *Blockchains* mais conhecidas atualmente, em que ambas têm o problema de não fornecer, por defeito, a proteção de dados e a sua consequente privacidade. O *Ethereum*, o principal foco desta dissertação, permite desenvolver Aplicações Descentralizadas (*DApps*) com a impossibilidade de estarem *offline* ou serem alvos de censura.

A União Europeia (EU) aprovou o Regulamento Geral sobre a Proteção de Dados (RGPD) em 2016, com penalizações apenas a serem aplicadas no dia 25 de Maio de 2018. Este regulamento é considerado um passo gigante para proteger a informação e os dados dos utilizadores, visto que este não afeta apenas organizações com escritórios na EU, mas também empresas no mundo todo que tenham clientes em territórios da União Europeia. Além disto, o regulamento estipula novas obrigações para organizações que manuseiam dados dos seus utilizadores, além de introduzir novos direitos para os mesmos, como o direito de apagamento dos dados. Este direito representa um desafio enorme para conseguir cumprir estritamente com o RGPD nas *DApps*, visto que as *Blockchains* como o *Ethereum* são, no seu design, imutáveis.

O trabalho desenvolvido nesta dissertação tenta cumprir com o RGPD e o seu direito problemático ao apagamento dos, ao desenvolver uma prova de conceito, uma *DApp* em *Ethereum*: *DFiles*, em que esta visa fornecer alguma maneira de proteger os dados dos seus utilizadores e também a sua privacidade. Além disto, também permite que os seus utilizadores submetam ficheiros encriptados além de os conseguirem desencriptar quando o seu *download* é efetuado. Foi também desenvolvida com uma metodologia *Agile*, com uma abordagem por iterações usando na maioria tecnologias descentralizadas, como por exemplo o *Interplanetary File System (IPFS)* e os contratos inteligentes do *Ethereum*, contando também com uma componente centralizada para efeitos de autenticação de utilizadores, ao mesmo tempo que adere à Engenharia de Desenvolvimento de Software para *Blockchain* (*BOSE*). Devido à complexidade do RGPD, apenas alguns dos seus aspetos foram selecionados para a sua implementação no *DFiles* como os direitos de apagamento dos dados, portabilidade, acesso e retificação.

O cumprimento do RGPD na *DFiles DApp* foi avaliado com recurso a uma análise estatística nos ficheiros encriptados e não encriptados submetidos pelos seus utilizadores, em que foram medidos o tempo gasto no seu *upload* e o custo total de transação em *Ethereum*, em ficheiros de quatro categorias diferentes: **pequenos** (1KB-1MB), **médios** (1MB-20MB), **grandes** (20MB-200MB) e **muito grandes** (200MB-2GB). No entanto, por limitações de *hardware*, esta análise estatística apenas foi feita para ficheiros até 14.2MB de tamanho. Pode ser concluído que os custos de transação para ficheiros não encriptados são ligeiramente

superiores, apesar deste aumento não ser significativo. Além disto, esta análise também concluiu que o tempo gasto nos ficheiros encriptados é substancialmente maior. Os dados dos ficheiros com tamanho superior a 14.2MB foram também registados. Ao compararmos os últimos dois registos dos ficheiros desencriptados, até 800MB de tamanho, concluímos que o seu tempo gasto é inferior aos útlimos dois registos para ficheiros encriptados até 14.2MB de tamanho.

Finalmente, pode-se concluir que encriptar ficheiros para cumprir com o direito ao apagamento de dados do RGPD é uma possível abordagem apenas para ficheiros pequenos e médios até 14.2MB de tamanho. A partir desta fase, e sem considerar limitações de *hardware*, os tempos gastos para *upload* de ficheiros encriptados tendem a aumentar exponencialmente. Assim sendo, o *Ethereum* e o *IPFS* têm obrigatoriamente que melhorar a sua tecnologia num futuro próximo para permitir novas e melhores técnicas de privacidade dos dados.

Recentemente, têm existido melhoramentos significativos no *Ethereum* e os seus contratos inteligentes que fazem com que o mundo do desenvolvimento de *DApps* se faça a um ritmo muito elevado. No futuro, o *Ethereum* poderá evoluir numa nova versão que poderá também trazer novos melhoramentos e controlos de privacidade que poderão permitir o cumprimento na totalidade do RGPD.

Acknowledgement

I would like to thank everyone that supported me throughout this amazing yet challenging year. First of all, without my family's support, this work would have not been possible. Secondly, a special appreciation goes to my supervisor, **Isabel Azevedo**, for her relentless support and dedication in ensuring I produced my best work this past year. She also guided me through every step of the way in learning Ethereum and its smart contracts. Without this outstanding support, suffice to say this work would have not been possible at all.

I would also to extend my gratitude to my dearest friends that provided an invaluable emotional support for me to never give up and continue pushing myself to be better, everyday. They are my brothers and sisters with whom I rely on my most difficult moments. A special thank you also goes to my friend, **Vitor Hugo Anjos**, who guided me when I was learning the difficult, yet awesome ReactJS frontend technology. He helped me solve many bugs in the code, in addition to instructing me how to best develop my frontend code throughout this work.

Finally, a word of appreciation to all the rock and roll bands out there, which provided me with beautiful music to never give up and always give my best. One such bands is AC/DC, which without this particular band, this work would have not been possible.
In rock we trust, work or bust!

Contents

List of Figures

List of Tables

List of Source Code

Glossary

backend The backend of an application, is essentially logic, or code, that runs on one or more servers. These receive and process requests from multiple clients. Most backend applications also have some form of persistent storage, with the most common being a database.

cryptoasset A cryptoasset refers to any cryptocurrency. Although they can be used as a form of money, they have other applications besides payment. For instance, cryptoassets can be used to pay for electricity and at the same time to purchase other coins which hold and intrinsic value.

cryptocurrency A Cryptocurrency (Cryptographic Currency) is the term used to describe digital currencies which utilize cryptography to secure their payment networks and transactions[1].

ERC20 ERC stands for "Ethereum Request for Comments" and is the official protocol for improvements proposal in the Ethereum Blockchain. 20 is the proposal ID.

escrow A bond, deed, or other document kept in the custody of a third party and taking effect only when a specified condition has been fulfilled[2].

fiat money A currency that a government has declared to be legal tender, but it is not backed by a physical commodity. The value of fiat money is derived from the relationship between supply and demand rather than the value of the material that the money is made of. Historically, most currencies were based on physical commodities such as gold or silver, but fiat money is based solely on the faith and credit of the economy[3].

frontend The frontend is typically a collection of technologies that display the complex logic provided from the backend to the end user. For instance, in a standard web application, there is code that fetches logic from the backend and is executed in the user's browser, with the final result of a beautifully generated user interface with backend logic.

genesis block A genesis block is the first block of a Blockchain. Modern versions of Bitcoin number it as block 0, though very early versions counted it as block 1. The genesis block is almost always hardcoded into the software of the applications that utilize its Blockchain. It is a special case in that it does not reference a previous block, and for Bitcoin and almost all of its derivatives, it produces an unspendable subsidy[4].

[1] https://steemit.com/cryptocurrency/@cryptoassets/what-are-crypto-assets-and-how-do-they-work

[2] https://en.oxforddictionaries.com/definition/escrow

[3] https://www.investopedia.com/terms/f/fiatmoney.asp

[4] https://en.bitcoin.it/wiki/Genesis_block

market capitalization Market capitalization is one way to rank the relative size of a cryptocurrency. It's calculated by multiplying the price by the circulating supply: Market Capitalization = Price *X* Circulating Supply[5].

maximum supply Maximum supply is the best approximation of the maximum amount of coins that will ever exist in the lifetime of the cryptocurrency[6].

mining Mining is essentially the process of validating transactions in a Blockchain.

premine A premine is where a developer allocates a certain amount of currency credit to a particular address before releasing the source code to the open community. This is often done based on the reasoning that they need to pay for certain features such as listing on exchanges and development of core features such as block explorers[7].

solidity A programming language mainly used for Ethereum Blockchain development.

token A form of digital money that can be used within a given system.

total supply Total supply is the total amount of coins in existence right now (minus any coins that have been verifiably burned)[8].

Turing complete A programming language that can solve any problem that a Turing machine can, with a finite amount of resources.

[5] https://coinmarketcap.com/faq/
[6] https://coinmarketcap.com/faq/
[7] https://www.cryptocompare.com/coins/guides/what-is-a-premine/
[8] https://coinmarketcap.com/faq/

List of Acronyms

AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
BOSE	Blockchain Software Engineering.
BTC	bitcoin.
CSS	Cascade Style Sheets.
CSV	Comma-seperated Values.
DAO	Decentralized Autonomous Organization.
DApp	Decentralized Application.
dBFT	Byzantine Fault Tolerant.
DLT	Distributed Ledger Technology.
DPIA	Data Protection Impact Assessment.
DPO	Data Protection Officer.
EOAs	Externally Owned Accounts.
ETH	Ether.
EU	European Union.
EVM	Ethereum Virtual Machine.
FEI	Front End Of Innovation.
GDPR	General Data Protection Regulation.
HOCs	Higher Order Components.
HTML	Hypertext Markup Language.
HTTP	HyperText Transfer Protocol.
HTTPS	HyperText Transfer Protocol Secure.
IPFS	Interplanetary File System.
JSON	JavaScript Object Notation.
LLL	Low-level Lisp-like Language.
MVC	Model View Controller.
NCD	New Concept Development Model.
NPPD	New Product and Process Development.

OM	Object Modelling.
OOP	Object Oriented Programming.
PKI	Public Key Infrastructure.
REST	Representational State Transfer.
RPC	Remote Procedure Call.
SPA	Software Product Assurance.
tesnet	test network.
UI	User Interface.
UML	Unified Modeling Language.
URL	Universal Resource Locator.
XML	Extensible Markup Language.
YAML	YAML Ain't Markup Language.

Chapter 1

Introduction

This chapter introduces the context, problem at hand and the main objectives described throughout this dissertation's work. It also details, briefly, the methodology behind it, in addition to several introductory concepts which are required to understand its importance. Finally, the document structure is also presented, where the contents of each chapter are summarized.

1.1 Context

Data privacy has been an important issue over the last few decades. Several big corporations have been known for mishandling and misusing data collected from individuals. Ensuring User privacy must be a priority in software applications, which was reinforced by the approval of the European Union (EU)'s General Data Protection Regulation (GDPR) in 2016 and currently enforceable. Compliance with this regulation is mandatory: penalties range from a minimum of 10 million Euros or 2% annual turnover (whichever is higher) to 20 million Euros or 4% annual turnover (whichever is higher) (European Union 2016).

In 1996, computer scientist and cryptographer Nick Szabo first conceived the idea of smart contracts (Szabo 1996). Generally, these are self-enforcing agreements (like real-world contracts) although expressed in computer code, which when executed, dictate the terms and conditions of the contract, without any of the parties involved trusting each other. Currently, they can be stored on a Blockchain; thus, they inherit its characteristic properties: immutable, public and decentralized. Smart contracts, once created and deployed, cannot be further altered or tampered with. The overall process for creating them must involve careful design, testing, and execution, or multiple unpatchable bugs and errors will potentially be exploited by attackers at some point in time. For example, the Decentralized Autonomous Organization (DAO) (Falkon 2017) and Parity hacks (Parity Technologies 2017) and (Petrov 2017) resulted in millions of US Dollars in stolen funds.

Two of the most well-known Blockchains are Bitcoin (Bitcoin Project 2018) and Ethereum (Ethereum Foundation 2018c). The former was created by an unknown person by the pseudonym of Satoshi Nakamoto, who pioneered the concept of digital currency, enforced by cryptography with the aim of solving the double spending problem. The latter is currently the biggest Blockchain to support smart contracts, which was specifically designed and created with an extended execution model for smart contracts (Ethereum Foundation 2018g). Ethereum also supports the development of Decentralized Applications (DApps)

which embrace its smart contracts and properties.

Note: throughout this document, bitcoin in lower and upper case will appear multiple times. When in lower case, it is often referred to bitcoin as a cryptocurrency. In the case it is written in upper case, it refers to the Bitcoin Blockchain and its ecosystem. The same can be applied to other Blockchains such as Ethereum and additional cryptocurrencies.

1.2 Problem Overview

A Blockchain is a distributed database of unchangeable records, at least in the light of the current computational power. It can also be a distributed ledger of all transactions that have been executed. There is a list of blocks, the Blockchain, where each block includes a list of transactions and a hash to the previous block, except for the genesis block, which is the same for all clients. Interactions with the Blockchain need a pair of private/public keys.

Under the GDPR (European Union 2016), individuals are entitled to two major rights: "right to erasure" and "right to rectification". For instance, since most Blockchains are immutable, how can Ethereum comply with these two individual rights? Further, can DApps be developed in compliance with the GDPR, even though acknowledging this issue?

In this work, the main problem at hand is to address how data protection regulations can be attended when using Ethereum Blockchain technology. To do this, a case study was developed: DFiles, a DApp built mainly with decentralized technologies such as the Interplanetary File System (IPFS) and Ethereum smart contracts in addition to adhering to Blockchain Software Engineering (BOSE). Here, answers to two important questions are attempted to be found by performing a statistical analysis:.

- Can DApps be fully compliant with the GDPR?
- Is it feasible to protect user data by encrypting it, in Ethereum (Homestead)?

1.3 Main Objectives and Goals

The main objectives and goals of this work are:

1. Analyze the Ethereum Blockchain technology and the European General Data Protection Regulation (GDPR)
2. Explore the use of Ethereum Blockchain technology in the light of regulatory concerns unleashed by the EU's GDPR and the necessity to, during the development of Ethereum Blockchain-based applications, ensure conformity to data protection laws
3. Propose a framework to enable data protection when using Ethereum Blockchain technology (the DFiles DApp)

In addition, due to the immutable nature of the Ethereum Blockchain, testing is **THE** most important part of this work. This goal can be described as follows:

- Test the solution, and identify its limitations and what can have conditioned the results, while identifying possible improvements
  - Some tests include:
    * DApp unit tests (with the Truffle Framework)
    * Smart contract static analysis with Securify and SmartCheck

1.4 Methodology

Data protection and its subset data privacy in the Ethereum Blockchain is a difficult problem to tackle today. An extensive knowledge of the existing privacy solutions is required for successfully developing a privacy-driven prototype with Ethereum smart contracts. For this reason, the first step was to analyze the multiple existing Ethereum data protection and privacy solutions currently available. Then, a comprehensive study was made to figure out if these solutions are in fact providing data privacy and protection to the public Ethereum Blockchain, in addition to understanding if the cost of achieving this is in fact feasible or not.

This is the **investigation phase**: an extensive search for existing data privacy and protection solutions was performed, to understand their strengths, weaknesses and if data protection and privacy is in fact feasible in Ethereum. However, note that only some of the most important solutions were explored. A more extensive study is required to be fully capable of delivering a cutting-edge data privacy and protection prototype in the public Ethereum Blockchain.

The other step in this work methodology was to define the **development** phase. In other words, this is how the project was planned and developed. The **Git** version control system was chosen to ensure an easy process of DApp development. Finally, the overall work schedule follows an Agile methodology: there are no fixed schedules for working in this project; instead all tasks were defined before hand on a weekly basis. The most important aspect was getting these tasks done, no matter if 2 hours were spent doing them or 20. This project largely benefited from this mentality; more work is done and the overall project quality in the end is substantially higher.

1.5 Introductory Concepts

This section includes some important albeit introductory concepts regarding this work which are later explored in the rest of this document.

Distributed Ledger Technology (DLT)

DLT is "a digital system for recording the transaction of assets in which the transactions and their details are recorded in multiple places at the same time. Unlike traditional databases, distributed ledgers have no central data store or administration functionality" (Bauerle 2017). Figure 1 compares a centralized ledger with a distributed one.

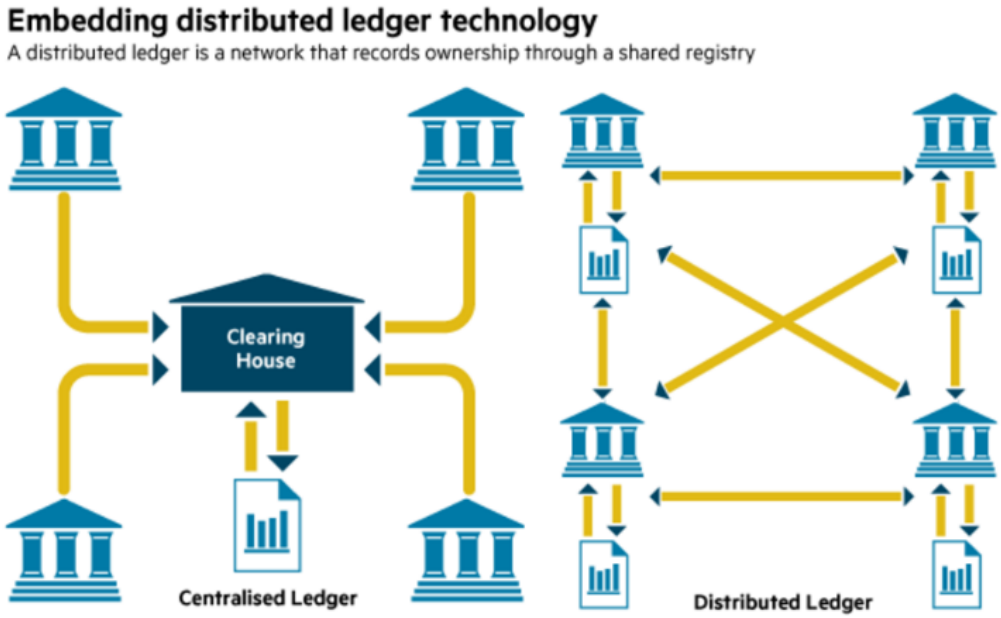

Figure 1: Two examples of ledgers: a centralized and a distributed one. From 10ZTalk.

A distributed ledger contains several nodes, which process and verify every item. Thus, they create consensus on each item's veracity by generating a record of each item (Rouse, Troy, and Pratt 2017). In other words, consensus can be defined as "coming to its own conclusions and then voting on those conclusions to make certain the majority agree with the conclusions" (Bauerle 2017). Moreover, distributed ledgers are used to record static (registry) and dynamic data (transactions) (Rouse, Troy, and Pratt 2017).
Once consensus is reached, all nodes maintain their identical copy of the ledger after it has been updated (Bauerle 2017). Figure 2 illustrates a distributed ledger.

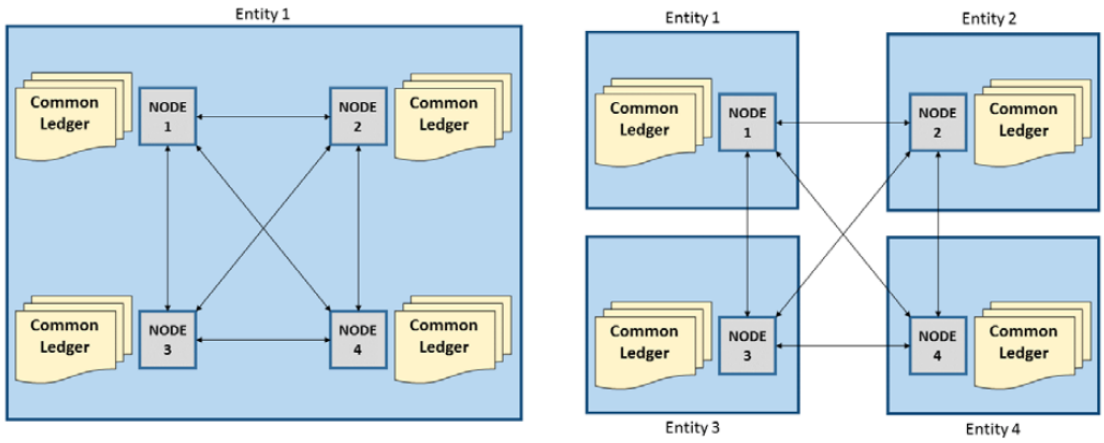

Figure 2: Distributed Ledger Technology. From True Economics.

Bitcoin and Ethereum Blockchains are the most noticeable and well-known implementations of a distributed ledger. These are important because they do not rely on a central authority

or middleman, thus making transactions a number of times faster with a fraction of the cost.

Moreover, a distributed ledger is a more transparent way of handling records and transactions due to its shared, immutable nature. Since every record is synced across the network, a successful cyber attack is highly unlikely.

The Ethereum Blockchain, as a distributed ledger, should be seen as a technology with the potential to touch and revolutionize every sector of today's society: from governments to financial institutions. Individuals could hold and better control their personal information and share these only when they decide to. Governments, for example, could better track digital property in a better and efficient way in addition to ensuring no third party can ever change a record because of censorship or oppression.

Encryption

The main purpose of encryption is to ensure data is kept secret, by transforming it (Miessler 2011). The goal here is not usability, rather it is "to ensure the data cannot be consumed by anyone other than the intended recipient(s).

Encryption transforms data into another format in such a way that only specific individual (s) can reverse the transformation. It uses a key, which is kept secret, in conjunction with the plain text and the algorithm, in order to perform the encryption operation. As such, the ciphertext, algorithm, and key are all required to return to the plain text" (Miessler 2011).

Encryption can be asymmetric or symmetric. The latter is a form of encryption when the key to both encrypt and decrypt is exactly the same.

On the other hand, asymmetric encryption uses two different keys: **one** to encrypt a string (a public key) while the other is used to decrypt it (a private key). The public key is made available for everyone to encrypt messages; however, only the intended recipient has access to the private key. This ensures that only him has the ability to decrypt messages.

Hashing

Hashing serves the task of ensuring integrity. For example, if something is changed — a file, a string of text — anyone can know it was in fact changed. Technically speaking, hashing takes arbitrary input and produces a fixed-length string that has the following attributes (Miessler 2011):

1. "The same input will always produce the same output
2. Multiple disparate inputs should not produce the same output
3. It should not be possible to go from the output to the input
4. Any modification of a given input should result in a drastic change to the hash

5. Hashing is used in conjunction with authentication to produce strong evidence that a given message has not been modified
6. This is accomplished by taking a given input, hashing it, and then signing the hash with the senders private key"

When a message is received and opened by the recipient, the senders public key can be used to validate the signature of the hash and then hash the message themselves. The final step is to compare it to the hash that was signed by the sender. In any event the two hashes match, it is an unmodified message, sent by the correct person.

What is a Blockchain?

The term Blockchain can be best described as a globally, duplicated and shared database or distributed ledger of transactions across multiple computers throughout the world. Because of the way consensus is obtained, without the need to trust any intermediate party — a Blockchain can be seen as a distributed ledger.

A Blockchain consists of a sequence of blocks where each block is built on its predecessors and contains information about transactions. The average transaction time varies from Blockchain to Blockchain.

Since a Blockchain is a distributed system, it faces many challenges such as scalability. It solves many problems as well, such as fixing the issue of multiple transactions being received out of order. Ethereum and Bitcoin Blockchains are implementations of a Blockchain.

Moreover, a Blockchain is essentially a combination of three technologies, as follows (Dannen 2017):

- "**Peer-to-peer networking:** A group of computers such as the BitTorrent network that can communicate among themselves without relying on a single central authority and therefore not presenting a single point of failure
- **Asymmetric cryptography:** A way for these computers to send a message encrypted for specific recipients such that anyone can verify the sender's authenticity, but only intended recipients can read the message contents. In Bitcoin and Ethereum, asymmetric cryptography is used to create a set of credentials for your account, to ensure that only you can transfer your tokens
- **Cryptographic hashing:** A way to generate a small, unique 'fingerprint' for any data, allowing quick comparison of large datasets and a secure way to verify that data has not been altered; in both Bitcoin and Ethereum, the Merkle tree data structure is used to record the canonical order of transactions, which is then hashed into a 'fingerprint' that serves as a basis of comparison for computers on the network, and around which they can quickly synchronize"

Smart Contracts

In the physical world, there are already financial contracts: "agreements to buy and sell at some point in the future, usually at a specified price. In the Ethereum context, smart contracts are agreements between accounts, to render a transfer of ether (that is, a payment) when certain conditions are met" (Dannen 2017).

These are called "smart" contracts due to the fact they are executed by a machine "and the assets (ether or other tokens) are moved automatically" (Dannen 2017). One interesting particularity of these smart contracts is their longevity. "These contracts could be enforced even hundreds of years after they've been written, assuming the network is still running then — and even if a lot of bad actors try to interfere" (Dannen 2017).

The Ethereum Virtual Machine (EVM) is where these contracts are run, in a "totally sandboxed and free from interference" (Dannen 2017) way in addition to being isolated from other networks. This last aspect makes it "impossible for a party to back out of a smart contract. In practical terms, this is because smart contracts are empowered to hold assets (ether or other tokens) in escrow and move them when the terms of the contract are met" (Dannen 2017).

Decentralized Applications

Internet users nowadays do not have sole control over the usage of their data in modern websites.

Decentralized Applications (DApps) can be thought as a "decentralized app store" where any developer can publish their applications which self-execute and are not controlled in any way by a third party or middleman.

There are many different ways to develop a **DApps** in different Blockchains. In this dissertation, the focus is primarily on **DApps** development in the Ethereum Blockchain.

Furthermore, **DApps** are protected from censorship, oppression, hacking or downtime due to the shared and secured nature of the Ethereum Blockchain.

There are multiple examples of existing **DApps** today. The State Of DApps Website[1] is dedicated to catalog most of all the existing and future DApps.

To sum up, one should think of a **DApps** as a software application that self-executes in a network which is immune to censorship, has no middleman and cannot be removed or erased as long as the network is up. These applications are considered the Web 3.0.[2]

[1] https://www.stateoftheDApps.com/

[2] https://www.coindesk.com/information/what-is-a-decentralized-application-dapp/

1.6 Document Structure

This document has the following structure:

- **Introduction** — chapter 1: in this chapter, the context of this dissertation's work is presented, in addition to the problem at hand and its main objectives. Further, the methodology behind this work is also described, followed by a brief explanation of basic concepts required to understand its importance
- **State of the Art** — chapter 2: the main idea here is to explore the different Ethereum Wallets in existence; understand the value of this work; evaluate the existing Blockchains and cryptocurrencies and point out their differences and similarities. Finally, the existing programming languages and frameworks to develop Decentralized Applications are studied here, in addition to their advantages and disadvantages
- **Value Analysis and Cryptoeconomics** — chapter 3: the main goal of this chapter is to evaluate how this work can bring value to the customer. These include aspects such as **value for the customer** and **perceived value**. In addition, the New Product and Process Development is presented here, combined with a thorough analysis of different Ethereum Wallets by applying the Analytic Hierarchy Process — the complete source code is located in appendices A and B. Last but not least, a quick study about how cryptocurrencies and their respective Blockchains are such a huge success in society in less than a decade, despite standard peer-to-peer networks such as BitTorrent can be considered a failure (both are peer-to-peer networks) — this is detailed in a section called Cryptoeconomics
- **The Ethereum Blockchain** — chapter 4: in this chapter, all the essential concepts regarding Ethereum and its token, Ether, are explained in addition to listing existing solutions based on Ethereum technology to protect personal data
- **Case Study** — chapter 5: DFiles, a Decentralized Application where users can upload and view their submitted files is elaborated here, as this dissertation's case study. It is developed mostly with decentralized technologies such as the Interplanetary File System and Ethereum smart contracts, in addition to a centralized component for user authentication. The first stage of the software development lifecycle is explained here, albeit adapted for DApps with Blockchain Software Engineering: DFiles' functional and non-functional requirements
- **Blockchain Software Engineering** — chapter 6: the main focus here is detailing the remaining stages of the software development lifecycle, in DFiles, while adhering to BOSE
- **Data Protection** — chapter 7: in this chapter, there are multiple important sections. First and foremost, a slimmed down version of the official GDPR[3] is presented. Secondly, a GDPR compliance plan for DApps in general is created in addition to documenting a specific one for the DFiles DApp, in addition to its implementation. Finally, the GDPR key changes regarding a previous legislation is explained

[3] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC

- **Evaluation** — chapter 8: here, answers to the following questions are provided:
  - Can a DApp be fully compliant with the GDPR?
  - Is encrypting data, such as files stored in IPFS nodes, feasible in the current version of the Ethereum Blockchain (Homestead)?

  To do this, a statistical analysis is performed on total elapsed time and transaction costs of the user's uploaded files on encrypted and unencrypted files to determine answers to the two questions mentioned above
- **Conclusions** — chapter 9: in this chapter, this dissertation's work is summarized in addition to presenting its multiple contributions. In addition to this, the multiple limitations encountered in DFiles are also explained with a future list of improvements (future work), closed by a personal overview about this work

Bibliography and references used in this document are placed after chapter 9.

Appendices can be found at the end of this document.

Chapter 2

State of the Art

Since the inception of the first cryptocurrency, bitcoin, developers and entrepreneurs alike have been developing new solutions in several Blockchains using different cryptocurrencies. What are the most important cryptocurrencies and Blockchains in the present and in the future? Moreover, what programming languages and frameworks are available to develop Decentralized Applications?
Finally, what kind of crypto wallet should be chosen when developing Ethereum Blockchain applications? Answers to most of these questions are present along this chapter:

1. **The Existing Blockchains and Cryptocurrencies** — section 2.1: the two most predominant Blockchains and their cryptocurrency are presented here: Bitcoin and Ethereum. The former due to the fact of being the first of its kind; the latter because it is the focus of this work. Furthermore, their economic values such as market capitalization, maximum supply and more are also described here. There are also other Blockchains and cryptocurrencies besides Ethereum and Bitcoin, which are also briefly detailed to ensure that a thorough knowledge of the different Blockchains and cryptocurrencies is acquired
2. **Tools and Technologies** — section 2.2: the different Ethereum tools and technologies are explored here, albeit generic. Other, more specific are described throughout this document. The more generic tools and technologies belong to three different types:
    - Wallets
    - Programming Languages
    - Ethereum Frameworks

2.1 Blockchains and Cryptocurrencies

Since the creation of the Bitcoin Blockchain, in recent years many Blockchains have risen from the aches. The purpose of this section is to explore the existing cryptocurrencies — in several factors such as **market capitalization**, **total supply**, **maximum supply**, **price in US$** and more — in addition to their Blockchains; the key differences from each other as well as their similarities.

Bitcoin

The Bitcoin Blockchain is the first one to surge and therefore often compared to. It has its own cryptocurrency, bitcoin. In addition to the general concept of what a standard Blockchain is, the Bitcoin Blockchain adds several important characteristics:

1. **Decentralized and Irreversible**: every transaction, once confirmed, in the Bitcoin Blockchain is irreversible and no central authority can control it - it is decentralized. This means that no central authority can actually revert any confirmed transaction, such as a bank. In addition, imagine a mad dictator comes into power in a specific country and tries to transfer money from every single citizen of that country to a specific account in his control. Due to the immutability of the Bitcoin Blockchain, this is no longer possible
2. **Anonymous and Transparent:** "users can hold multiple bitcoin addresses which are not linked to any person's name, address, phone number, bank account, or any other personally identifying information" (Coindesk 2015)
3. **Ultra Low Transaction Fees:** transferring from one bitcoin address to another costs fractions of a dollar. In contrast with, say, the Western Union, where the fees are substantially higher
4. **Secure:** "Bitcoin funds are locked in a public key cryptography system. Only the owner of the private key can send cryptocurrency. Strong cryptography and the magic of big numbers makes it impossible to break this scheme" (Rosic 2017b)
5. **Permissionless:** anyone can use cryptocurrency. It is just a software that everybody can download for free. After installing it, anyone can receive and send bitcoin or other cryptocurrencies[1]
6. **Fast Peer-To-Peer Payments:** transactions usually take, on average, minimum 10 minutes to a maximum of approximately 3000 minutes[2]. Compare this to standard centralized transactions, it takes on average minimum days, maximum months...
7. **Low Processing Fees:** this is usually true, however due to the current price of 1BTC = US $6,293.27 , transaction fees have increased dramatically: one bitcoin transaction can take as much as 40% of the total transaction value (!)[3], thus making it unfortunately almost on pair to standard banking solutions
8. **Solves The Double Spending Problem:** "double-spending is the result of successfully spending some money more than once. Bitcoin users protect themselves from double spending fraud by waiting for confirmations when receiving payments on the Blockchain, the transactions become more irreversible as the number of confirmations rises" (Bitcoin Wiki 2018a)

In addition to this, the Table 1 provides further details of the importance of this cryptocurrency.

[1] https://blockgeeks.com/guides/what-is-bitcoin/
[2] https://Blockchain.info/charts/avg-confirmation-time
[3] https://www.cnbc.com/2017/12/19/big-transactions-fees-are-a-problem-for-bitcoin.html

Table 1: Bitcoin data courtesy of **Coin Market Cap**.

Bitcoin		
ID	**Symbol**	**Cryptocurrency Rank**
bitcoin	BTC	1
Price In US$	**Market Capitalization in US$**	
$6,293.27	$109,003,983,294.00	
Maximum Supply	**Total Supply**	
21,000,000	17,320,725	

Ether and its Ethereum Blockchain

Ethereum is a decentralized software platform with its Decentralized Applications (DApps) and Smart Contracts. The latter are built without any forms of censorship, downtime or fraud. It runs on the EVM, in addition to being Turing complete. It also has its own token, Ether (ETH).

The Ethereum Blockchain will be the main focus of this dissertation; in the upcoming chapters, everything regarding this Blockchain will be explained - with special emphasis in chapter 4.
Last but not least, Table 2 explores the differences between the cryptocurrencies of both the Ethereum and Bitcoin Blockchains.

Table 2: Bitcoin VS Ether, from **One Month**.

	bitcoin (BTC)	**Ether (ETH)**
What Is It?	A currency	A token
Supply Style	Deflationary (a finite # of bitcoin will be made)	Inflationary (much like fiat money, where more tokens can be made over time)
Supply Cap	21 million in total	18 million every year
Smallest Unit	1 Satoshi = 0.00000001 BTC	1 Wei = 0.000000000000000001 ETH
New token issuance time	Every 10 minutes approximately	Every 10 to 20 seconds
Amount of new token at issuance	12.5 at the moment. Half at every 210,000 blocks	5 per every new block
Utility	Used for purchasing goods and services, as well as storing value (much like how we currently use gold)	Used for making Decentralized Applications (DApps) on the Ethereum Blockchain
Price	Around $6,293.27 at the moment	Around $199.52 at the moment
Purpose	A new currency created to compete against the gold standard and fiat currencies	A token capable of facilitating Smart Contracts (For example: a lawyer's contract, an exchange of ownership of property, and voting)

Finally, some further information about Ether in Table 3.

Table 3: Ethereum data courtesy of **Coin Market Cap**.

Ethereum		
ID	**Symbol**	**Cryptocurrency Rank**
ethereum	ETH	2
Price In US$	**Market Capitalization in US$**	
$199.52	$20,461,752,355.00	
Maximum Supply	**Total Supply**	
None	102,553,878	

Besides Ethereum and Bitcoin, there are other cryptocurrencies and Blockchains. These are explored in H.

2.2 Tools and Technologies

This section describes the existing, generic tools and technologies in Ethereum. As previously mentioned, these are from three types:

- Wallets
- Programming Languages
- Frameworks

Ethereum Wallets

Choosing a good Ethereum Wallet is the first step anyone should do. But what exactly is an Ethereum Wallet? It is, in short, "a gateway to decentralized applications on the Ethereum Blockchain. It allows you to hold and secure ether and other cryptoassets built on Ethereum, as well as write, deploy and use smart contracts" (Ethereum Foundation 2018c).There are multiple wallets available; the objective of this section is to explore the different ones available, and point out their key differences and similarities:

- **ETHAddress**[4] is an open source paper wallet: both private and public keys are printed on paper. It is the cheapest form of physical storage available, a big advantage towards keeping a wallet safe. Table 4 provides a quick insight on ETHAddress.

Table 4: ETHAddress quick information, available at: **Coin Sutra**.

Supported Platforms	Web Paper Wallet
Privacy	No registration or personal information needed
Smart Contracts Support	No

- **MyEtherWallet**[5] has one key advantage comparative to other Ethereum wallets: it does not rely on any server, therefore storing Ethereum's private key on one's machine. It is fully open source and allows smart contracts to be accessed and written.

  On top of that, "it has an inbuilt BTC to ETH (and vice-versa) swap facility" (Khatwani 2018).Table 5 provides a quick overview on MyEtherWallet.

Table 5: MyEtherWallet quick information, available at: **Coin Sutra**.

Supported Platforms	Web Wallet, Chrome Extension
Privacy	No registration or personal information needed
Smart Contracts Support	Yes

[4] https://github.com/ryepdx/ethaddress.org
[5] https://www.myetherwallet.com/

- **Parity** is defined by Parity Technologies (2018) as "the fastest and most secure way of interacting with the Ethereum Blockchain. Our client powers much of the infrastructure of the public Ethereum network and is used by companies and users alike."

  The best feature of parity is undoubtedly its ease-of-use. In addition, it can be thought as having more features than any other Ethereum Wallet:

  - **For Users:**
    - "Account, address book and multi-sig management
    - Key creation, importing and exporting
    - Web3 Dapp browser
    - Hardware and electronic cold wallet support
    - Name registry support
  - **For Developers:**
    - DApp browser and registration
    - ERC20 balances and registration
    - Solidity development environment
    - Clean, modular codebase
  - **For Enterprise:**
    - Fast transaction processing
    - Proof-of-Authority consensus engines
    - Privacy and control features variety of deployment solutions
    - Ability to augment features" (Parity Technologies 2018)

  Table 6 offers a very simplified insight on parity.

  Table 6: Parity quick information available at: **Coin Sutra**.

Supported Platforms	Mac, Linux and Windows
Privacy	No registration or personal information needed
Smart Contracts Support	Yes

  On a final note, parity has had its fair share of severe problems: a $280m US Parity Ethereum Wallet was frozen due to a hack[6], thus alerting everyone to the possibility of more hacks and its lack of security. If this did not happen, parity would be the number one choice for Ethereum development.

- **Mist**[7] is the official Ethereum Wallet; it is secured via a strong password. The private keys are stored on the device; there are no servers that mist depends on. On top of

[6] https://www.theregister.co.uk/2017/11/10/parity_280m_ethereum_wallet_lockdown_hack/
[7] https://github.com/ethereum/mist/releaseshttps://github.com/ethereum/mist/releases

this, Mist also supports the **Remix IDE**[8] for quick Solidity development and testing and has **ShapeShift**[9] built-in for quick cryptocurrency exchange.

The only **BIG** downside is the Ethereum Blockchain synchronization size comparatively to other Ethereum Wallets such as Parity: 3-4 times larger.

A quick glance regarding MIST can be seen in the Table 7, below.

Table 7: MIST quick information, available at: **Coin Sutra**.

Supported Platforms	Mac, Linux and Windows
Privacy	No registration or personal information needed
Smart Contracts Support	Yes

- Metamask is one of the best Ethereum Wallets, due to the following features, according to their YouTube video[10]:
  - **Metamask** is encrypted, locally stored in the browser — no account information touches their servers
  - In Metamask, it is quite easy to send Ether from one account to another
  - Instead of having an Ethereum node running on a machine, one connects to their servers via their Google Chrome or Firefox extension — Metamask accesses the Ethereum Blockhain via the **Web3JS API**[11]
  - Support, out-of-the-box, for multiple Ethereum Networks:
    * **Main Ethereum Network**
    * **Ropsten Test Network**
    * **Kovan Test Network**
    * **Rinkeby Test Network**
    * **Localhost 8545**
    * **Custom RPC**

A short table — Table 8 — summarizing Metamask can be found below.

Table 8: Metamsk quick information, Available at: **Coin Sutra**.

Supported Platforms	Firefox and Chrome extensions
Privacy	No registration or personal information needed
Smart Contracts Support	No

[8] https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.19+commit.c4cbbb05.js
[9] https://shapeshift.io/#/coins
[10] https://www.youtube.com/watch?v=6Gf_kRE4MJU
[11] https://web3js.readthedocs.io/en/1.0/

Ethereum Programming Languages

In addition to choosing an Ethereum Wallet, there is also the need to understand the different programming languages used to develop Ethereum Blockchain Decentralized Applications — most commonly known as the Web 3.0. Here it is explored these programming languages in addition to examining the advantages and disadvantages of each one. These include:

- **Solidity**: "a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the EVM. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features" (Ethereum Foundation 2018f).

  It is one of the official programming languages for developing smart contracts in the Ethereum Blockchain, combined with the MIST Ethereum Wallet. It has many unique features to any other Ethereum programming language, with the best one being the **REMIX IDE**[12], where anyone can develop and debug smart contracts in a very easy, simple and comfortable manner.

  Finally, note that most smart contracts are written in this language, which makes it the undisputed choice for developing Decentralized Applications with smart contracts in the Ethereum Blockchain. Every other programming language will be compared to Solidity

- **Vyper**: a an experimental language, with the following goals and advantages regarding other Ethereum programming languages (Ethereum Foundation 2018h):
  - "**Security** — it should be possible and natural to build secure smart contracts in Vyper **Language and compiler simplicity** — the language and the compiler implementation should strive to be simple
  - **Auditability** — Vyper code should be maximally human-readable. Furthermore, it should be maximally difficult to write misleading code. Simplicity for the reader is more important than simplicity for the writer, and simplicity for readers with low prior experience with Vyper (and low prior experience with programming in general) is particularly important"

  Vyper surely has potential to become a serious contestant to the already king of Ethereum programming languages, Solidity, with its new simplistic and secure mentality. Table 9 provides a quick comparison between Vyper and Solidity.

[12] https://remix.ethereum.org/

Table 9: Comparing Vyper and Solidity Ethereum programming languages.

	Vyper	**Solidity**
Documentation	Poor, early stages[13]	Widely available[14]
Support	Alpha software, little support	Most widely supported Ethereum Programming language
Community	Alpha software, very few people actually are active in developing with Vyper	Vibrant and enthusiastic
Features	Only essential features are present; the aim here is **auditability**	The most feature-rich Ethereum programming language
Influenced by (Programming Languages)	Python	JavaScript, C++ and Python

- **Serpent** is an assembly language that compiles to EVM code "that is extended with various high-level features. It can be useful for writing code that requires low-level opcode manipulation as well as access to high-level primitives" (Buterin 2017).

  Serpent is widely similar to the Python programming language, and has had great documentation and support until recently. Now fully replaced by its newer, better version known as Solidity, developing in Serpent is discouraged: "being a low-level language, Serpent is **NOT RECOMMENDED** for building applications unless you really really know what you're doing. The creator recommends Solidity as a default choice, Low-level Lisp-like Language (LLL) if you want close-to-the-metal optimizations, or Viper if you like its features though it is still experimental" (Buterin 2017).

Ethereum Frameworks

After choosing the Ethereum Wallet and programming language, the third and final step is to pick the best Ethereum Framework for DApps development.
Three of these frameworks include:

- **Truffle:** is a development environment, testing framework and asset pipeline for Ethereum, "aiming to make life as an Ethereum developer easier. With Truffle, you get: Built-in smart contract compilation, linking, deployment and binary management" (Consensys 2018)

  Truffle offers the following features:

  - "Built-in smart contract compilation, linking, deployment and binary management
  - Automated contract testing with Mocha[15] and Chai[16]
  - Configurable build pipeline with support for custom build processes

[15] https://mochajs.org/
[16] http://chaijs.com/

  - Scriptable deployment & migrations framework. Network management for deploying to many public & private networks
  - Interactive console for direct contract communication
  - Instant rebuilding of assets during development
  - External script runner that executes scripts within a Truffle environment"

- **Embark:** is a framework which allows anyone to "easily develop and deploy Decentralized Applications (DApps).
  Embark currently integrates with EVM blockchains (Ethereum), Decentralized Storages (IPFS), and Decentralized communication platforms (Whisper and Orbit)" (Status Research & Development GmbH 2018).

  It has the following features (Status Research & Development GmbH 2018):

  - "Automatically deploy contracts and make them available in your JavaScript code. Embark watches for changes, and if you update a contract, Embark will automatically redeploy the contracts (if needed) and the dapp
  - Contracts are available in JavaScript with Promises
  - Do Test Driven Development with Contracts using Javascript
  - Keep track of deployed contracts; deploy only when truly needed
  - Manage different chains (e.g testnet, private net, livenet)
  - Easily manage complex systems of interdependent contracts"

- **Meteor:** Meteor is a full-stack JavaScript platform for developing modern web and mobile applications. "Meteor includes a key set of technologies for building connected-client reactive applications, a build tool, and a curated set of packages from the Node.js and general JavaScript community" (Meteor Development Group Inc. 2018).

  According to the Ethereum Foundation (2018d), Meteor is a perfect fit for developing DApps:

  - "It is purely written in JavaScript and has all the tools a Software Product Assurance (SPA) needs (Templating engine, Model, on-the-fly compiling, bundling)
  - A development environment, which has live reload, Cascade Style Sheets injection and support for many pre-compilers (LESS, CoffeeScript, etc) out of the box
  - All frontend code as single index.html with one js and css file plus your assets, using meteor-build-client
  - It embraces full reactivity, which make building consistent interface much easier (similar to angular.js $scope or binding)
  - It has a great model called Minimongo, which gives you a mongoDB like interface for a reactive in-memory database, which can also be auto-persisted to local storage or indexedDB"

  However, meteor provides an improvement on a centralized problem: syncing a backend and a frontend. This contradicts the purpose of DApps: applications that are

decentralized. For this reason alone, meteor should only be used on very special circumstances.

Embark VS Truffle

Both Embark and Truffle are well suited for Ethereum Decentralized Applications development. However, one should note their key differences before choosing to stick with one particular framework. According to Reddit user dmpldr (2017), the main distinct aspects of these two frameworks are:

1. **Embark:**
   - Embark is fully committed to "trust-less" applications; this means that one cannot build an end-to-end hybrid application (e.g. the inclusion of a nodejs server communicating with smart contracts)
   - Developing Web front-end solutions is very easy to do; this is mainly to the fact that Embark exposes the ethereumjs API via Javascript Promises and keeps track on local code changes and testnet
   - If a DApps involves backend servers, the convenient features mentioned in the point above are not present out-of-the-box
2. **Truffle:** the main difference versus Embark is that it has no entanglement with the contract consumers; this means that Truffle can run tests independently

Chapter 3

Value Analysis and Cryptoeconomics

Understanding the potential of cryptocurrencies and their Blockchains is fundamental in 2018 and not possible without the notion of value, perceived value and value for the customer — value analysis — in addition to understanding the idea of benefits and penalization for any Blockchain users: cryptoeconomics. This chapter focuses on these two aspects.

3.1 Value Analysis

Since Satoshi Nakamoto first conceived the Bitcoin Blockchain, its interest has increased exponentially: people value more and more Bitcoin and other Blockchains. For this reason, it is important to define the very key concept of value, which can be:

"Value has been defined in different theoretical contexts as need, desire, interest, standard criteria, beliefs, attitudes, and preferences" (Nicola, E. P. Ferreira, and J. J. P. Ferreira 2012).

When developing any great product, the customer should always come first. In terms of this work, the customer could be defined as the **Ethereum community** or more specifically: **the Ethereum community in countries where the European Union enforce laws**. Recall that the main focus of this dissertation is **to propose a framework to enable the use of Ethereum Blockchain technology in conformity to the GDPR**: the product. In terms for value for the customer, this translates to **a free, secure and reliable product which complies to the EU's data protection regulation**, despite having present all the limitations[1] of the Ethereum Blockchain.

Last but not least, there should be the notion that **perceived value** "is the consumers's overall assessment of the utility of a product based on perceptions of what is received and what is given" (Zeithaml 1988). This is better translated into Table 10.

[1] https://www.coindesk.com/information/blockchains-issues-limitations/

Table 10: "Conceptualising value for the customer: an attributional, structural and dispositional analysis" (Woodall 2003).

Benefits		Sacrifices
Attributes	**Outcomes**	
Perceived Quality	Functional Benefits	Price
Product Quality	Utility	Market Price
Quality	Use Function	Monetary Costs
Service Quality	Aesthetic Function	Financial
Technical Quality	Operational Benefits	Costs
Functional Quality	Economy	Costs Of Use
Performance Quality	Logistical Benefits	Perceived Costs
Service Performance	Product Benefits	Search Costs
Service	Strategic Benefits	Acquisition Costs
Service Support	Financial Benefits	Opportunity Costs
Special Service Aspects	Results For The Customer	Delivery and Installation Costs
Additional Services	Social Benefits	Costs Of Repair
Core Solution	Security	Training and Maintenance Costs
Customization	Convenience	Non-monetary Costs
Reliability	Enjoyment	Non-financial Costs
Product Characteristics	Appreciation From Users	Relationship Costs
Product Attributes	Knowledge, Humour	Psychological Costs
Features	Self-expression	Time
Performance	Personal Benefits	Human Energy
	Association With Social Groups	Efforts
	Affective Arousal	

Table 10 portrays a list of example benefits and sacrifices for the customer, in terms of utility of a product. The Ethereum Blockchain and its smart contracts, pave the way to endless opportunities and demonstrate best the notion of these benefits and sacrifices in relation to monetary benefits and penalties, as better described in section 3.2, **cryptoeconomics**.

On top of that, Table 11 can be presented to demonstrate in a quick and elegant way a brief notion of **perceived value** for the customer, in terms of this work.

Table 11: Perceived value in the context of this work.

Scope \ Domain	PRODUCT	SERVICE	RELATIONSHIP
BENEFIT	Built on the Ethereum Blockchain	Responsiveness	Image
	As Fast as possible	Flexibility	Trustless
	Trustless	Reliability	Immutable
	Open Source	Bug Hunting and Testing	Anonymous
	Immutable	Anonymous	As Fast as Possible
	Compliant With EU's Data Protection Regulation	Trustless	Valuable
	Cryptographically Secure	Dedicated Support	Important
	Fully Tested		
	Theoretically Hack-proof		
	Always Online		
SACRIFICE	Time/effort/energy		
	Conflit		
	Beta/Alpha State		
	Lack Of Features		
	Not For Everyone		

Table 11 illustrates the contrast between benefits and sacrifices for the final product of this dissertation's work: **a framework that enables the use of the Ethereum Blockchain in compliance with the European General Data Protection Regulation**. This product has the following benefits for the customer:

1. **Built on the Ethereum Blockchain:** taking part of the decentralized revolution of the next decade, with all its advantages! Key features of the Ethereum Blockchain include:
   - **Trustless:** the client will never learn about the identity of another person using the framework; therefore trustless communications
   - **Immutable:** once a transaction is made in the framework, because of its Blockchain nature, it is impossible to alter it
   - **Cryptographically Secure:** all the latest standards of cryptography are present in the framework, courtesy of the Ethereum Blockchain; thus making transactions highly secure
   - **Theoretically Hack-proof:** due to its distributed nature, the Ethereum Blockchain makes it highly unlikely and extremely difficult to hack the framework, although there is a slim chance of it happening

- **No Downtime; Always Online:** the Ethereum Blockchain, due to its many nodes connected, establishes the certainty of always online distributed applications

2. **As Fast as Possible:** although significantly slower than traditional centralized architectures such as client-server, the Ethereum Blockchain will improve its speed exponentially over the next few years; so will the framework
3. **Open Source:** the framework is open sourced[2]
4. **Compliant With EU's General Data Protection Regulation:** rest assure this framework is compliant with the most recent general data protection laws
5. **Fully Tested:** with the help of the Ethereum and open source communities, the framework should be fully tested

As for the service, it can be defined as the **software solution support (the framework) between the author of this dissertation and the customers — as previously described**. This includes the following benefits for the customer:

1. **Responsiveness:** once a problem is found, and due to the framework's open sourced nature, the client should get immediate feedback
2. **Flexibility:** all the feedback given to the client should be done based on his schedule
3. **Reliability:** all the feedback given should be reliable
4. **Bug Hunting and Testing/Fixing:** once a bug is found, it should be fixed as fast as possible
5. **Anonymous:** due to the properties of the Ethereum Blockchain, it should be as anonymous as possible
6. **Trustless:** same as above, however getting fixes done should be done regardless if the client is trustworthy or not
7. **Dedicated Support:** there should be dedicated support for the client; thus making the framework better each day

The last piece of the puzzle between the customer and this author is the relationship between them:

1. **Image:** the client should know competent people are behind the framework
2. **Trustless:** same concept as above
3. **Immutable:** same as before
4. **Anonymous:** already mentioned
5. **Valuable:** the client should feed appreciated and valued when testing the framework
6. **Important:** the client should feed the importance of testing and using the framework

Finally, every product (this one included) has its own sacrifices to the client:

1. **Time/effort/energy:** since the framework is free and open sourced, the major downside to the client is the time, effort and energy spent while testing it

[2] https://bitbucket.org/duarte_1110199/dfiles-ethereum-dapp/src/master/

2. **Conflict:** sometimes the client may not like how the development of the framework is going, which may lead to conflict
3. **Beta/Alpha State:** in the early stages, the framework might be so unstable and unusable that discourages the client from trying it
4. **Lack of Features:** same thing as the point above, in addition to having very little features in the beginning of development
5. **Not for Everyone:** although the Ethereum community might understand and appreciate the framework, its use is limited due to its complex and experimental nature

Fuzzy Front End, New Product and Process Development (NPPD) and Commercialization

When developing a new product, framework or idea, the innovation process, according to Koen (2004), can be defined as: "the innovation process may be divided into three parts — Figure 3.

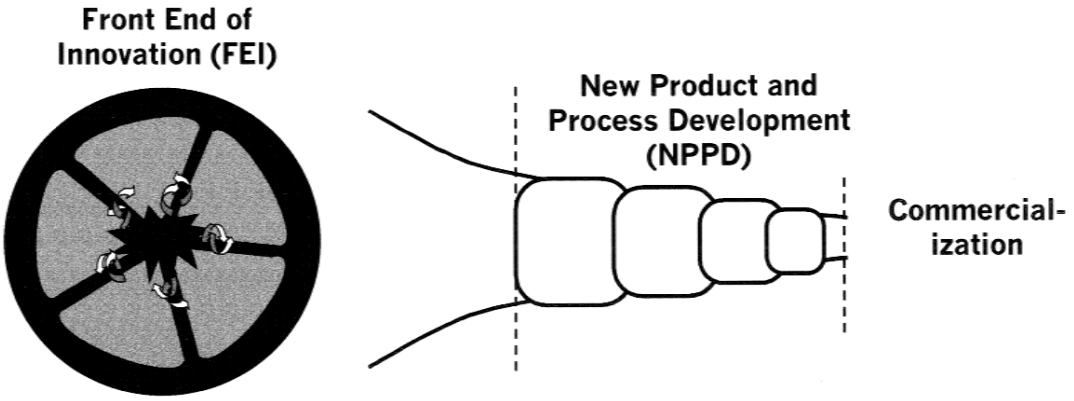

Figure 3: The entire innovation process.

- **The Fuzzy Front End (FFE) or Front End of Innovation (FEI):** activities that come before the formal and well structured New Product and Process Development Portion — less structured and less predictable
- **The New Product and Process Development Process (NPPD):** structured with a formalized and prescribed set of activities and questions
- **Commercialization**"

New Concept Development Model (NCD)

The **NCD**: "provides a common language and definition of the key components of the Front End of Innovation"(Koen 2004). This model can be seen in Figure 4.

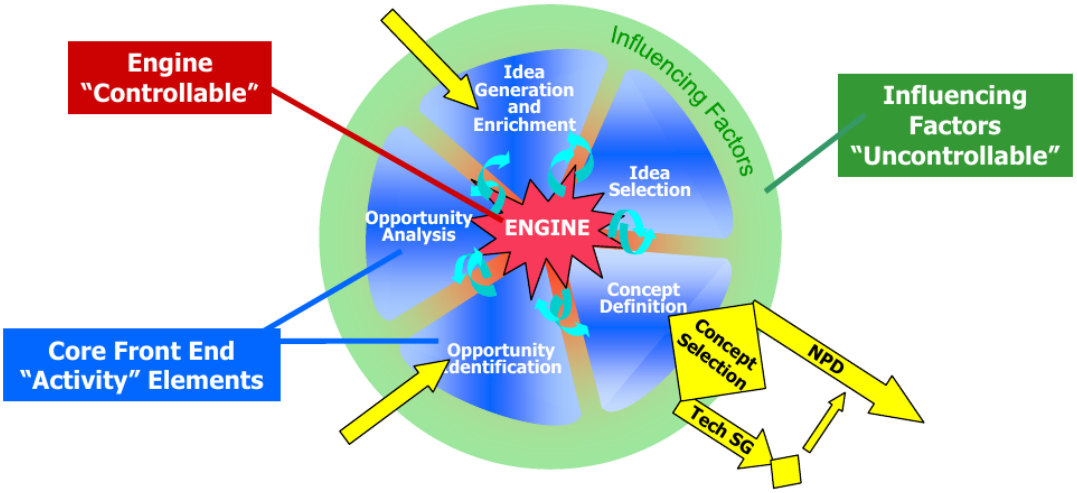

Figure 4: New Concept Development Model from (Koen et al. 2001).

The NCD has three main parts (Koen 2004):

1. "The inner area defines the five key elements comprising the Front End Of Innovation (FEI).
2. The Engine or 'bull's-eye' portion which drives the five front-end elements and is fueled by the leadership and culture of the organization
3. The Influencing Factors, or environment on the periphery, consists of Organizational Capabilities, Business Strategy and the Outside World"

In addition, it is important to consider the five elements of the frontend (Koen 2004):

1. **Opportunity Identification:** "this is where the organization, by design or default, identifies the opportunities that the company might want to pursue. Business and technological opportunities are explicitly considered so that resources will eventually be allocated to new areas of market growth and/or operating effectiveness and efficiency. This element is typically driven by the goals of the business. For example, the opportunity may be a near-term response to a competitive threat, a breakthrough possibility for capturing competitive advantage, or a means to simplify/speed-up/reduce the cost of operations. The opportunity could be an entirely new direction for the business or a minor upgrade to an existing product. It could also be a new product platform, a new manufacturing process, a new service offering, or a new marketing or sales approach
2. **Opportunity Analysis:** this is where the organization, by design or default, identifies the opportunities that the company might want to pursue. Business and technological opportunities are explicitly considered so that resources will eventually be allocated to new areas of market growth and/or operating effectiveness and efficiency. This element is typically driven by the goals of the business. For example, the opportunity may be a near-term response to a competitive threat, a breakthrough possibility for capturing competitive advantage, or a means to simplify/speed-up/reduce the cost of operations. The opportunity could be an entirely new direction for the business or a minor upgrade to an existing product. It could also be a new product platform, a new manufacturing process, a new service offering, or a new marketing or sales approach

3. **Idea Genesis:** genesis is the birth, development and maturation of the opportunity into a concrete idea. This represents an evolutionary process in which ideas are built upon, torn down, combined, reshaped, modified,and upgraded. The idea may go through many iterations and changes as it is examined, studied, discussed, and developed. Direct contact with customers/users and linkages with other cross-functional teams, as well as collaboration with other companies and institutions, often enhance this activity. Idea Genesis may be a formal process including brainstorming sessions and idea banks so as to provoke the organization into generating new or modified ideas for the identified opportunity. A new idea may also emerge outside the bounds of any formal process — an experiment that went awry, a supplier offering a new material, or a user making an unusual request. Idea Genesis may feed Opportunity Identification, demonstrating that the NCD elements may proceed in a non-linear fashion — advancing and nurturing ideas and opportunities wherever they occur. The output of this element is typically a more completely developed description of the 'sensed' idea or product concept

4. **Idea Selection:** in most businesses there are so many product/process ideas that the critical activity is to choose which ideas to pursue in order to achieve the most business value. Selection may be as simple as an individual's choice among many self-generated options or as formalized as a prescribed portfolio method. More formalized project selection and resource allocation in the FEI is difficult due to the limited information and understanding at this point. Definition of the financial return in the FEI is at best often just a 'wild' guess. Better selection models specifically designed for the FEI are needed so that market and technology risks, investment levels, competitive realities, organizational capabilities, and unique advantages, along with financial returns, may all be considered. Idea Selection, as in Opportunity Analysis, should be less rigorous than in the NPPD since many ideas must be allowed to grow and advance with less certainty

5. **Concept and Technology Development:** the final element of the model involves the development of a business case based on estimates of market potential, customer needs, investment requirements, competitor assessments, technology unknowns, and overall project risk. The level of formality of the business case varies according to the nature of the opportunity (e.g., new market, new technology and/or new platform), level of resources, organizational requirements to proceed to the NPPD and the business culture (formal, informal or hybrid). In some organizations, this is considered the initial stage (i.e., Stage 0) of the NPPD process"

The next step is applying the five frontend elements mentioned above to this dissertation's work (propose a framework to enable the use of Ethereum Blockchain technology in conformity to the European Union's General Data Protection Regulation):

1. **Opportunity Identification** occurred when the European Union decided to enforce data privacy rules to any EU citizen or any organization making business in the EU; some of these rules do not comply with the way the Ethereum Blockchain is implemented and designed

2. **Opportunity Analysis** took place when the European Union drafted and passed a law enforcing data privacy laws for its citizens. This raises many questions such as:

    - What are the consequences of **NOT** complying with these data privacy rules in the EU?

- Is there any possible scenario where enforcing these data privacy rules is not mandatory?
- Can the Ethereum Blockchain be an exception to these rules?
- What are the costs of complying with these rules?

These questions raise important questions regarding the feasibility of cooperating with the data protection regulation. After further consideration about these rules, a sensible option is to comply with the European Union

3. In **Idea Genesis**, several methods were considered to comply with the data privacy laws:
   - Use of private Blockchains
   - Use of traditional distributed peer-to-peer networks
   - Development of a new framework to enable the use of Ethereum Blockchain technology in conformity with the European Union and its general data protection regulation
   - Continue playing safe by using traditional cloud services such as **Microsoft Azure**[3]
4. In **Idea Selection**, all of the above ideas were considered; with the clear winner being the development of a new framework to enable the use of Ethereum Blockchain technology in conformity with the general data protection regulation by the European Union. There are many factors to this; the most important one being the opportunity to acquire cutting-edge knowledge and possibly be prepared for the Blockchain revolution in the near future
5. Finally, in **Concept and Technology Development**, a new software development processes was started to develop the already mentioned framework

The Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) "is a method for multi-criteria decision making that breaks the problem down based on decision criteria, subcriteria, and alternatives that could satisfy a particular goal. The criteria are compared to one another, the alternatives are compared to one another based on how well they comparatively satisfy the subcriteria, and then the subcriteria are examined in terms of how well they satisfy the higher-level criteria" (Radziwill 2016).

In order to make a decision in the Analytic Hierarchy Process, it must be decomposed in the following four steps (Saaty 2008):

1. "Define the problem and determine the kind of knowledge sought
2. Structure the decision hierarchy from the top with the goal of the decision, then the objectives from a broad perspective, through the intermediate levels (criteria on which subsequent elements depend) to the lowest level (which usually is a set of the alternatives)

[3] https://azure.microsoft.com/en-us/

3. Construct a set of pairwise comparison matrices Each element in an upper level is used to compare the elements in the level immediately below with respect to it
4. Use the priorities obtained from the comparisons to weigh the priorities in the level immediately below. Do this for every element. Then for each element in the level below add its weighed values and obtain its overall or global priority. Continue this process of weighing and adding until the final priorities of the alternatives in the bottom most level are obtained"

Recall from section 2.2 the existing Ethereum Wallets. The goal of this part is to apply the AHP to **determine the best Ethereum Wallet** based on the following criteria:

- **Features:** all wallets will be evaluated based on their features; from the most feature-rich to the least
- **Security:** all wallets will have their security tested; an Ethereum Wallet can claim they are secure, however, if any past hack occurred, their preference will degrade substantially
- **Performance:** all wallets are tested for their speed in connecting to the Ethereum Blockchain
- **Privacy:** this will evaluate if all wallets respect a user's privacy such as encryption and private key storage on device, or if they do support such things
- **Storage:** since most wallets connect to the Ethereum Blockchain, a grade will be given based total disk space they occupy

Two diagrams of the AHP overall process can be seen below — figures 5 and 6 —, one complete and another the simplified version.

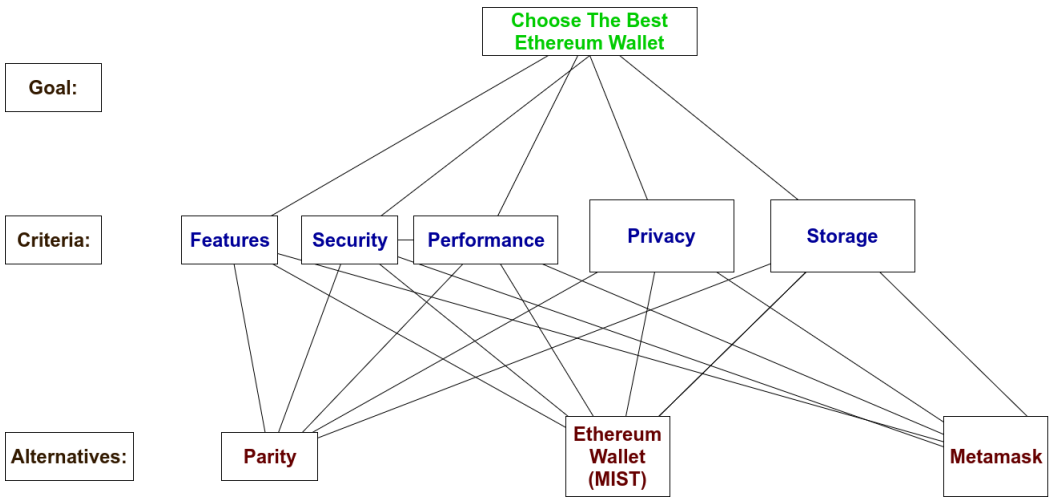

Figure 5: AHP application diagram — complete.

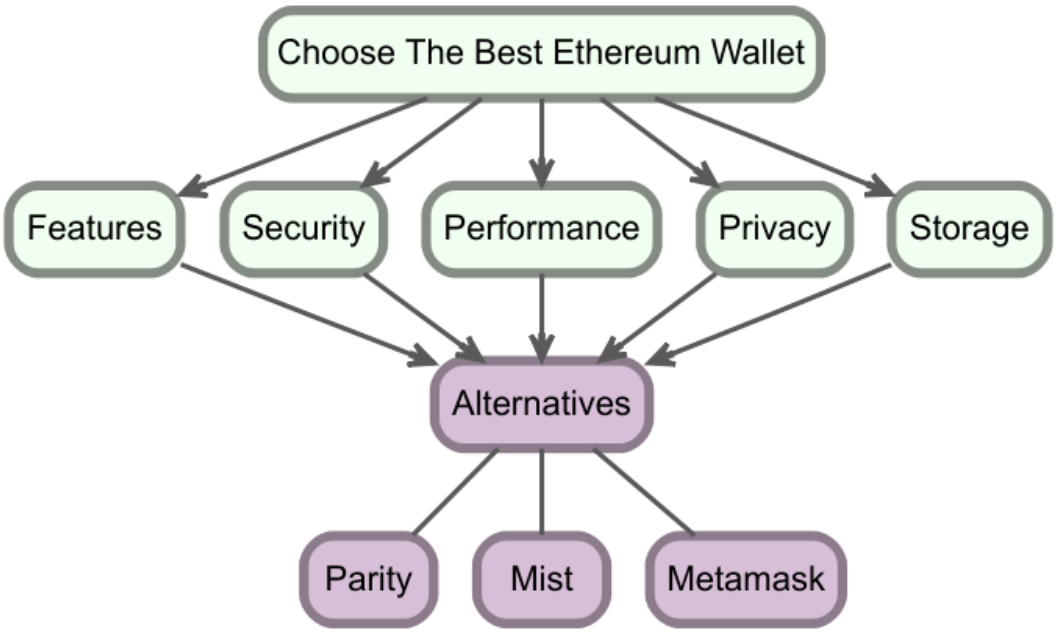

Figure 6: AHP application diagram — simplified

As for the alternative Ethereum Wallets, these include:

1. **Parity**
2. **Ethereum Wallet (MIST)**
3. **Metamask**

Pairwise Comparisons

Each node (rectangle in the hierarchy — see figures 5 and 6) will be derived from a series of measurements: pairwise comparisons involving all the nodes.

The next step is to complete step 3 from item 3.1, which is to compare each level, two by two, regarding to their contribution to the nodes directly above them. The results are then entered into a matrix with the goal of assigning priorities for all the nodes on the level.

In this case, there are three Ethereum Wallets, or alternatives (Parity, Ethereum Wallet (MIST) and Metamask). These are then compared each one with the remaining two.
Then, pairwise comparisons are made with respect to each criterion: **Parity VS Ethereum Wallet (MIST)**, **Parity VS Metamask** and **Ethereum Wallet (MIST) vs Metamask**. For each comparison, it will be judged which member of the pair is weaker regarding the criterion under evaluation. Then, a relative weight is assigned to the other Ethereum Wallet.

The AHP fundamental scale is used when assigning weights, Table 12.

Table 12: The fundamental scale for pairwise comparisons (Saaty 2008).

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
2	Week or slight	
3	Moderate importance	Experience and judgement slightly favor one activity over another
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favor one activity over another
6	Strong plus	
7	Very strong or demonstrated importance	An activity is favored very strongly over another; its dominance demonstrated 5 in practice
8	Very, very strong	
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
Reciprocals of above	If activity i has one of the above non-zero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i	A reasonable assumption
1.1-1.9	If activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of the small numbers would not be too noticeable, yet they can still indicate the relative importance of the activities.

Alternatives VS Criteria

Using available knowledge about these Ethereum Wallets, AHP will be used to develop a scale that measures their relative strengths regarding the criteria (by comparing pairs of Ethereum Wallets):

- **Features**
- **Security**

- **Performance**
- **Privacy**
- **Storage**

Finally, by using the AHP fundamental scale (Table 12), a weight is assigned to each of the criteria mentioned above to the other Ethereum Wallet.

On top of this, each criterion is also compared in pairwise operations. For example, security is compared to privacy, storage, performance and features. The same is applied for every other criterion. This is presented in the complete YAML Ain't Markup Language (YAML) Appendix B and R code in Appendix A.

Note: in the following paragraphs, the criteria that is compared will have an additional text-based table, which only provides further explanation about their comparisons and does not count directly for the AHP process.

Features

Regarding features, a summarized comparison is best displayed in Table 13.

Table 13: AHP alternatives compared with respect to features.

Alternatives compared with respect to features				
Parity	**4**	**Ethereum Wallet (MIST)**	**1**	*
Parity	**7**	**Metamask**	**1**	**
Ethereum Wallet (MIST)	**9**	**Metamask**	**1**	***

*: Parity has an impressive array of features, such as a Web3 Dapp browser, hardware and electronic cold wallet support, fast transaction processing. MIST, on the other hand, has the REMIX IDE integrated, which is a great advantage versus Parity. Both support smart contract development. Since Parity has significantly more features comparative to MIST, it is moderately preferred.
**: Metamask has the biggest advantage of the company themselves running an Ethereum Blockchain node, which frees significant storage from a person's computer. Both have Dapp browsing and are cross-platform. However, Metamask only supports Firefox and Google Chrome as an extension. For this reason, Parity is very strongly preferred.
***: The biggest advantage of the Ethereum Wallet is that it's officially supported and has a wide array of features, such as the built-in REMIX IDE for smart contract development. As such, MIST is extremely preferable.

The next step is to follow steps three and four from item 3.1. In other words, the remaining goals are to create a set of pairwise comparison matrices and use these to construct weight matrices - where priorities have been assigned a specific weight in a given matrix. These weight matrices, for each Ethereum Wallet, are listed in Table 14.

Table 14: AHP weights matrix — features.

Features	Parity	Ethereum Wallet (MIST)	Metamask
Parity	1	4	7
Ethereum Wallet (MIST)	1/4	1	9
Metamask	1/7	1/9	1

Security

As for security, a summarized comparison can be seen in the Table 15.

Table 15: AHP alternatives compared with respect to security.

Alternatives compared with respect to security				
Parity	**1**	**Ethereum Wallet (MIST)**	**9**	*
Parity	**1**	**Metamask**	**9**	**
Ethereum Wallet (MIST)	**1**	**Metamask**	**1**	***

*: Parity has outstanding security, as does MIST. However, there was a **very damaging hack**[4] regarding Parity, which makes one reconsider preferring Parity over anything else. For this reason, MIST is extremely preferred.
**: The devastating hack mentioned above is the sole reason for not considering Parity secure at all, as this can easily happen in the future.
***: Metamask has their servers, while MIST is secured on a person's device. Both implement extremely secure measures, such as strong passwords. For this reason, they are equal.

Moreover, the weights matrix for security can be found in Table 16.

Table 16: AHP weights matrix — security.

Security	Parity	Ethereum Wallet (MIST)	Metamask
Parity	1	1/9	1/9
Ethereum Wallet (MIST)	9	1	1
Metamask	9	1	1

Performance

Talking about performance, a summarized comparison can be seen in Table 17.

Table 17: AHP Alternatives compared with respect to performance.

Alternatives compared with respect to performance				
Parity	**6**	**Ethereum Wallet (MIST)**	**1**	*
Parity	**1**	**Metamask**	**3**	**
Ethereum Wallet (MIST)	**1**	**Metamask**	**9**	***

[4] https://www.theregister.co.uk/2017/11/10/parity_280m_ethereum_wallet_lockdown_hack/

***:** MIST has substantially higher startup and sync (with the Ethereum Blockchain) times. For this reason alone, Parity is strongly preferred.
****:** Metamask is slightly preferred regarding performance due to the fact that it does not have the need to sync with the Ethereum Blockchain locally. Everything is done in Metamask's servers.
*****:** Metamask destroys MIST for the same reason above. Metamask is of extreme importance.

The next step is determining the weights matrix for performance — Table 18.

Table 18: AHP weights matrix — performance.

Performance	Parity	Ethereum Wallet (MIST)	Metamask
Parity	1	6	1/3
Ethereum Wallet (MIST)	1/6	1	1/9
Metamask	3	9	1

Privacy

Regarding privacy, a summarized comparison can be seen in Table 19.

Table 19: AHP alternatives compared with respect to privacy.

Alternatives compared with respect to privacy				
Parity	**1**	**Ethereum Wallet (MIST)**	**1**	*
Parity	**1**	**Metamask**	**1**	**
Ethereum Wallet (MIST)	**1**	**Metamask**	**1**	***

,** and **: all wallets employ strong password rules and allow the user to access easily their Ethereum Wallet private keys. For this reason, all wallets are equal.

In addition, the weights matrix for privacy is defined — Table 20.

Table 20: AHP weights matrix — privacy.

Privacy	Parity	Ethereum Wallet (MIST)	Metamask
Parity	1	1	1
Ethereum Wallet (MIST)	1	1	1
Metamask	1	1	1

Storage

Regarding storage, a summarized comparison can be seen in Table 21.

Table 21: AHP alternatives compared with respect to storage.

Alternatives compared with respect to storage				
Parity	**7**	**Ethereum Wallet (MIST)**	**1**	*
Parity	**1**	**Metamask**	**9**	**
Ethereum Wallet (MIST)	**1**	**Metamask**	**9**	***

*: Parity Ethereum Blockchain size is about 10GB, versus MIST which is about 30-50GB. Mist is therefore, strongly preferred.
**: Metamask does not sync with the Ethereum Blockchain on a persons's device; therefore requires no storage. Metamask is extremely preferred.
***: Metamask is extremely preferred for the reason above.

Finally, the storage weights matrix — Table 22.

Table 22: AHP weights matrix — storage.

Storage	Parity	Ethereum Wallet (MIST)	Metamask
Parity	1	7	1/9
Ethereum Wallet (MIST)	1/7	1	1/9
Metamask	9	9	1

Making The Decision

In order to decide which Ethereum Wallet is best, all of the weight matrices were applied in an AHP external tool, written in the R programming language: **Analytical Hierarchy Process (AHP) with R**[5]. The final result can be seen in Figure 7.

	Weight	Metamask	Mist	Parity	Inconsistency
Choose The Best Ethereum Wallet	100.0%	49.6%	26.7%	23.8%	11.9%
Privacy	39.3%	13.1%	13.1%	13.1%	0.0%
Security	22.9%	10.9%	10.9%	1.2%	0.0%
Storage	20.9%	16.5%	1.0%	3.5%	37.6%
Performance	13.6%	9.0%	0.8%	3.8%	4.6%
Features	3.3%	0.2%	1.0%	2.2%	26.3%

Figure 7: AHP final result for best Ethereum Wallet.

Metamask is the best Ethereum Wallet with a score of 49.6%, following by Parity with a score of 26.7% and in third the MIST Wallet with 25.8%.
Regarding the most important criterion, this is **privacy** with a score of 39.3% and the least important one being **features** with a score of only 3.3%.

Note: the complete source code of the R file used in the AHP matrices calculation in addition to its correspondent YAML file (where the alternatives, criteria and goal are specified) is available in appendices A and B.

[5] https://github.com/gluc/ahp

3.2 Cryptoeconomics

Cryptoeconomics, as the name might suggest, is a combination of two English words: **cryptography and economics** which are the focus of this section.

Cryptography "is associated with the process of converting ordinary plain text into unintelligible text and vice-versa. It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Cryptography not only protects data from theft or alteration, but can also be used for user authentication" (The Economic Times 2018).

Modern cryptography concerns with (The Economic Times 2018):

- "**Confidentiality:** information cannot be understood by anyone
- **Integrity:** information cannot be altered
- **Non-repudiation:** sender cannot deny his/her intentions in the transmission of the information at a later stage
- **Authentication:** sender and receiver can confirm each"

There are three types of cryptographic techniques used in general[6]:

1. **Symmetric Cryptography**
2. **Hash Functions**
3. **Public-key Cryptography**

In Blockchain technology, there are several cryptographical functions for its operations. Below are listed some of its main functions[7]:

- **Hashing**
- **Signatures**
- **Proof Of Work**
- **Zero Knowledge Proofs**

The "Failure" of Traditional Peer-To-Peer Networks

The most common traditional peer-to-peer network is undoubtedly BitTorrent[8]. It allows anyone to download files via the BitTorrent protocol with programs as BitTorrent clients. Files are distributed across all users who are downloading or have downloaded one. BitTorrent breaks down each file into very small chunks of data, which makes sharing (that is, uploading) it possible after only a few minutes — a process known as seeding. Users can also choose to stop seeding a file or do it with very reduced speeds: there's a bandwidth

[6] https://economictimes.indiatimes.com/definition/cryptography
[7] https://blockgeeks.com/guides/what-is-cryptoeconomics/
[8] http://www.bittorrent.com/

variable in question here. An example of a file being shared via BitTorrent can be seen in Figure 8.

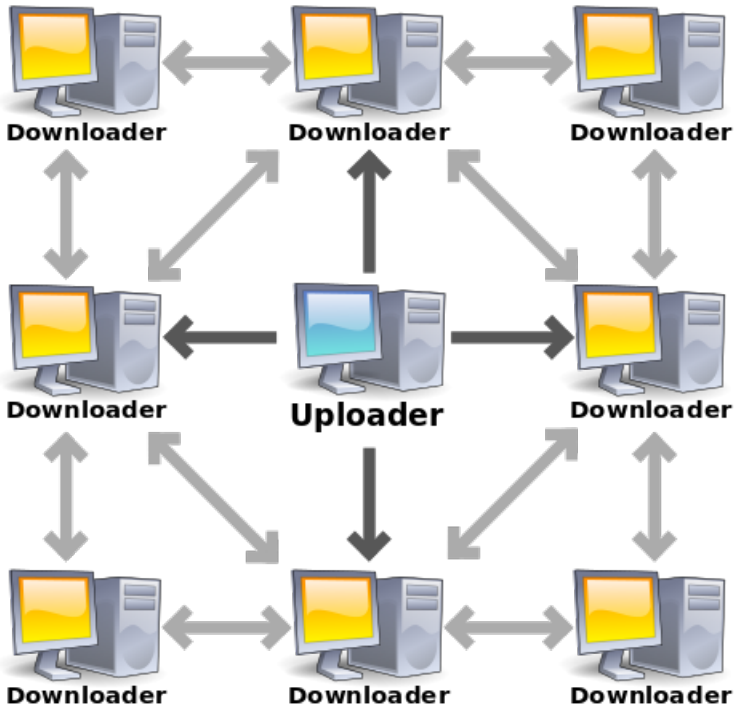

Figure 8: A user downloading a file from the BitTorrent network from Tech-infected.

With the knowledge of how BitTorrent works, the question at hand here why can traditional peer-to-peer networks be considered a failure?
The most interesting approach to this question is by asking another: what incentive does a user have to, after downloading a file, continue seeding it for anyone in the network? Besides, why **should** anyone keep seeding when there is no financial gain whatsoever?

This is the main problem of traditional peer-to-peer networks such as BitTorrent. Without a financial incentive, human beings are less likely to pursue a certain technology such as BitTorrent. There were cases, however, of popular peer-to-peer technologies that relied on donated bandwidth and power from users such as **Folding At Home**[9] for the PlayStation 3 back in the day. Once more, without a financial incentive, Sony was forced to discontinue this[10].

Fast forward to post 2009, when Satoshi Nakamoto created the Bitcoin Blockchain and its array of financial incentives and punishments. In the next parts, it will be explored how exactly did Satoshi Nakamoto capture the hearts and wallets of many human beings with this punishment/reward system.

[9] http://folding.stanford.edu/
[10] https://www.theverge.com/2012/10/22/3537502/sony-removing-ps3-folding-home

Economics

Blockchains differ from traditional peer-to-peer systems due to the fact that they provide its users with economic incentives to get work done. In a standard solid economic system, there should be rewards for people that get work done in addition to a punishment system for miners who do not act ethically or fail to do a good job. In the following subsections, one will learn how a Blockchain incorporates all of these economic fundamentals.
Blockchain participants have the following two incentive sets:

Incentive set #1 (Rosic 2017c):

- **Tokens:** "The actors who actively participate and contribute to the Blockchain get assigned cryptocurrencies for their efforts
- **Privileges:** Actors get the decision-making rights which gives them the right to charge rent"

Incentive set #2 (Rosic 2017c):

- **Rewards:** "Good participants get a monetary reward or decision-making responsibility for doing well
- **Punishments:** Bad participants have to pay a monetary fine or they have their rights taken away for behaving badly"

Cryptocurrencies and Value

Cryptocurrencies have value and trust the same way traditional money has. When a commodity is trusted and given a value, it becomes a currency like gold or fiat money. In order to understand the value of a given commodity, one must learn about one of the oldest rules of economics: **supply and demand**.

Supply and demand is the "relationship between the quantity of a commodity that producers wish to sell at various prices and the quantity that consumers wish to buy. The price of a commodity is determined by the interaction of supply and demand in a market. The resulting price is referred to as the equilibrium price and represents an agreement between producers and consumers of the good. In equilibrium the quantity of a good supplied by producers equals the quantity demanded by consumers" (Encyclopedia Britannica 2018) — Figure 9.

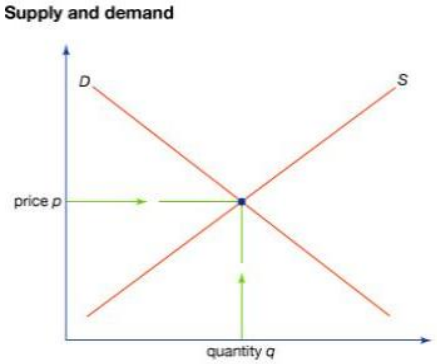

Figure 9: "Illustration of the relationship of price to supply (S) and demand (D)" (Encyclopedia Britannica 2018).

The Demand Curve

The quantity of a commodity demanded depends on the "price of that commodity and potentially on many other factors, such as the prices of other commodities, the incomes and preferences of consumers, and seasonal effects. In a basic economic analysis, all factors except the price of the commodity are often held constant; the analysis then involves examining the relationship between various price levels and the maximum quantity that would potentially be purchased by consumers at each of those prices. The price-quantity combinations may be plotted on a curve, known as a demand curve, with price represented on the vertical axis and quantity represented on the horizontal axis. A demand curve is almost always downward-sloping, reflecting the willingness of consumers to purchase more of the commodity at lower price levels. Any change in non-price factors would cause a shift in the demand curve, whereas changes in the price of the commodity can be traced along a fixed demand curve" (Encyclopedia Britannica 2018) — Figure 10.

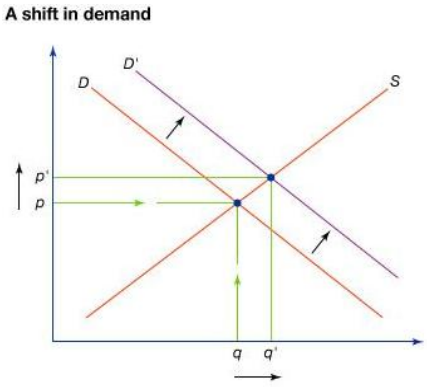

Figure 10: "Illustration of an increase in equilibrium price (p) and equilibrium quantity (q) due to a shift in demand (D)" (Encyclopedia Britannica 2018).

The Supply Curve

The quantity of a commodity that is supplied "in the market depends not only on the price obtainable for the commodity but also on potentially many other factors, such as the prices

of substitute products, the production technology, and the availability and cost of labor and other factors of production. In basic economic analysis, analyzing supply involves looking at the relationship between various prices and the quantity potentially offered by producers at each price, again holding constant all other factors that could influence the price. Those price-quantity combinations may be plotted on a curve, known as a supply curve, with price represented on the vertical axis and quantity represented on the horizontal axis. A supply curve is usually upward-sloping, reflecting the willingness of producers to sell more of the commodity they produce in a market with higher prices. Any change in non-price factors would cause a shift in the supply curve, whereas changes in the price of the commodity can be traced along a fixed supply curve" (Encyclopedia Britannica 2018) — Figure 11.

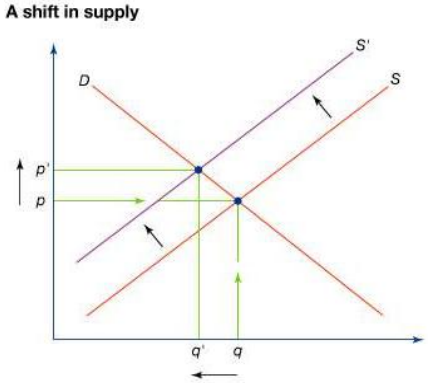

Figure 11: "Illustration of an increase in equilibrium price (p) and a decrease in equilibrium quantity (q) due to a shift in supply (S)"(Encyclopedia Britannica 2018).

Market Equilibrium

It is the function of a market to "equate demand and supply through the price mechanism. If buyers wish to purchase more of a good than is available at the prevailing price, they will tend to bid the price up. If they wish to purchase less than is available at the prevailing price, suppliers will bid prices down. Thus, there is a tendency to move toward the equilibrium price. That tendency is known as the market mechanism, and the resulting balance between supply and demand is called a market equilibrium.

As the price rises, the quantity offered usually increases, and the willingness of consumers to buy a good normally declines, but those changes are not necessarily proportional. The measure of the responsiveness of supply and demand to changes in price is called the price elasticity of supply or demand, calculated as the ratio of the percentage change in quantity supplied or demanded to the percentage change in price. Thus, if the price of a commodity decreases by 10 percent and sales of the commodity consequently increase by 20 percent, then the price elasticity of demand for that commodity is said to be 2.

The demand for products that have readily available substitutes is likely to be elastic, which means that it will be more responsive to changes in the price of the product. That is because consumers can easily replace the good with another if its price rises. The demand for a product may be inelastic if there are no close substitutes and if expenditures on the

product constitute only a small part of the consumer's income. Firms faced with relatively inelastic demands for their products may increase their total revenue by raising prices; those facing elastic demands cannot.

Supply-and-demand analysis may be applied to markets for final goods and services or to markets for labor, capital, and other factors of production. It can be applied at the level of the firm or the industry or at the aggregate level for the entire economy" (Encyclopedia Britannica 2018).

In terms of bitcoin, its supply is fixed at 21 million: market capitalization for all bitcoins. Due to its total limit, there must be regulations — which apply to all cryptocurrencies — in place to make sure bitcoins are progressively harder to mine; otherwise miners will mine all the remaining bitcoins at once, which results in decreased overall worth. These regulations include[11]:

- A new block is added to the chain every 10 minutes which leads to a reward of 25 bitcoins
- The difficulty of the bitcoin protocol is constantly being increased

Cryptocurrencies Demand

Many factors are at play regarding demand for cryptocurrencies (Rosic 2017c):

- "What is the history of the currency?
- Has it been subject to a hack lately?
- Does it consistently generate results?
- How good is the team behind it?
- Does it have potential to become better?
- How much is the hype around it?

All these factors determine how 'hot' the currency is and as a result, the value shifts depending on its demand".

Game Theory and Keeping Miners Honest

Keeping an unregulated, decentralized peer-to-peer system and its users is a tough challenge. To ensure miners obey the rules, one can look no further to game theory in economics: "Game theory is the study of human conflict and cooperation within a competitive situation. In some respects, game theory is the science of strategy, or at least the optimal decision-making of independent and competing actors in a strategic setting" (Investopedia Staff 2016). In other words, game theory is the study of strategic decision-making — making decisions which benefit you the most, while at the same time keeping tabs of the

[11] ttps://blockgeeks.com/guides/what-is-cryptoeconomics/

competitor's decisions. A fundamental component of game theory is the **Nash Equilibrium**.

Nash Equilibrium

A Nash Equilibrium is a state where "a party takes the most optimal strategy keeping in mind the actions of the other party and they cannot gain anything by changing around their strategy" (Rosic 2017c).

	B Takes Action	B Does Not Take Action
A Takes Action	(4,4)	(4,0)
A Does Not Take Action	(0,4)	(0,0)

Table 23: An example of the Nash Equilibrium — the "Payoffs Matrix" from (Rosic 2017c).

In the Table 23, two people are presented: **A** and **B**. The numbers are units of payoffs that a person will get after taking or not taking action.

There are two possibilities for each person[12]: they take action or they do not take action. Starting with the best strategy for person B:

- **A takes action:** B has a payout of 4 if he takes action and 0 if he does not. Obviously, the best approach for B is to take action
- **A does NOT take action:** B has a payout of 0 for not taking action and 4 for taking action
- **Conclusion:** regardless of the action performed by A, the best strategy for B is to take action

As for person A:

- **B takes action:** A has a payout of 4 if he takes action, and 0 if he does not take action. The best way is for A to take action
- **B does not take action:** A has a payout of 4 if he takes action and 0 if he does not
- **Conclusion:** regardless of what B does, the best way forward for A is to take action

The best way forward for both is to take action. Thus, the Nash Equilibrium is as highlighted in the following table:

	B Takes Action	B Does Not Take Action
A Takes Action	(4,4)	(4,0)
A Does Not Take Action	(0,4)	(0,0)

Table 24: The Nash Equilibrium: both A and B take action from (Rosic 2017c)

[12] https://blockgeeks.com/guides/what-is-cryptoeconomics/

Any implementation of the Blockchain self-imposes the Nash Equilibrium. Figure 12 explains this.

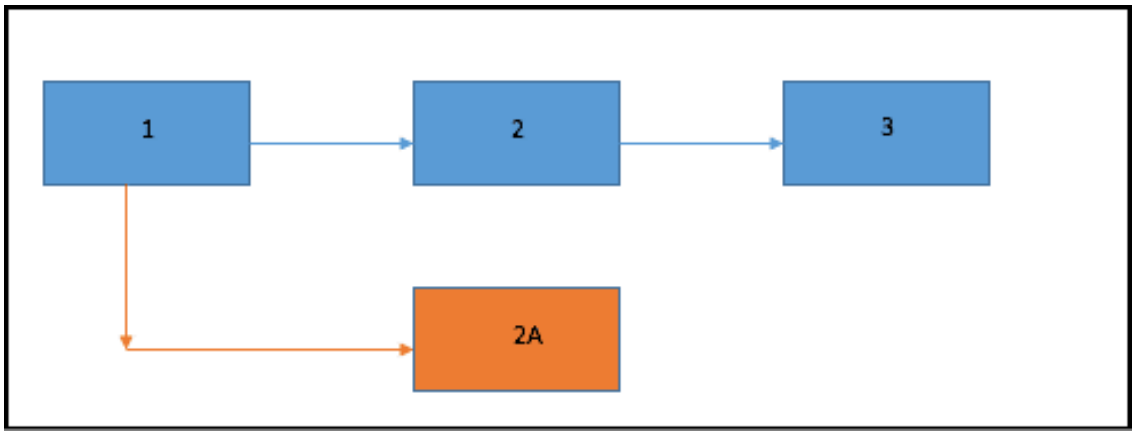

Figure 12: An example of the Nash Equilibrium being imposed in a Blockchain from (Rosic 2017c).

The above Blockchain has three blocks in blue which are part of main chain: **1**, **2** and **3**; another one **2A**, orange. Consider the following scenario: a malicious miner mines block **2A** and attempts to hardfork for its personal financial gain. Is it possible for other miners to join him on the newly created block and split the gains? The answer is no, as "any block that is mined on an individual block is not considered a valid block" (Rosic 2017c). The other miners will ignore this block and continue mining for valid ones. Since there is not value associated to this invalid block, why should anyone continue to support it? In addition to this, consider that mining is a very expensive process in terms of total electricity cost.

Consider the second scenario: "what if a lot of miners decide to join the new miner and mine on the new block?
The problem with that is that the Blockchain network is a huge and widely distributed network wherein communication and coordination is next to impossible" (Rosic 2017c). For this reason, a coordinated attack is next to impossible. Miners will simply choose the best route for maximum profits - this is how the Nash Equilibrium in the main chain is maintained[13].

Punishment in a Blockchain

As with any efficient economic systems, "good actions should be rewarded and negative actions should be punished. How does punishment work in a game theory model" (Rosic 2017c)? Consider Table 25, a payoff matrix where "the payoff for the participants is high but the implication on the society, in general, is very high" (Rosic 2017c).

	B Does Not Commit A Crime	B Commits A Crime
A Does Not Commit A Crime	(1,1)	(1,4)
A Commits A Crime	(4,1)	(4,4)

Table 25: Punishment in the Blockchain: Payoff Matrix from (Rosic 2017c).

[13] https://blockgeeks.com/guides/what-is-cryptoeconomics/

Think about this scenario: two people, A and B, are both about to commit a crime. By applying the same guidelines as before, the Nash Equilibrium results in both people committing a crime. In today's society, however, this is not the case. Why is this the case? Due to the concept of punishment.

Consider this: when someone commits a crime, they get punished with -5. Table 25 above changes to Table 26.

	B Does Not Commit A Crime	B Commits A Crime
A Does Not Commit A Crime	(1,1)	(1,-1)
A Commits A Crime	(-1,1)	(-1,-1)

Table 26: Punishment in the Blockchain: Payoff Matrix part 2 from (Rosic 2017c).

The Nash Equilibrium here is (1,1), which means that both people A and B do not want to commit a crime. Introduce the ideology that society gets punished as well if a crime is committed, then society has an incentive to keep everyone at check.

In a Blockchain, "any miners who are not following the rules and mining illegal blocks are punished by having their privileges taken away and risk social ostracization. By using simple game theory and punishment system, the miners are kept honest" (Rosic 2017c).

Chapter 4

The Ethereum Blockchain

As explained before, cryptocurrencies and their respective Blockchains have had until now a major role in shifting significantly towards decentralization, security and immutability of information. Ethereum is no stranger to this; it is the second most important Blockchain after Bitcoin. It allows for the development of Decentralized Applications (DApps) with popular frameworks such as Truffle and Embark.

Furthermore, Ethereum is a complex Blockchain: there are concepts that require one's attention and explanation. These include, for example, EVM and smart contracts. Additionally, there is the ongoing issue of data protection and privacy in the Ethereum Blockchain, as every transaction is set public.

The purpose of this chapter is to further examine and explain all the Ethereum concepts required for anyone to understand the core of this work, in addition to exploring the different data protection and privacy solutions in the Ethereum Blockchain.

4.1 The Ethereum Virtual Machine

According to the Ethereum Foundation (2018i), at the heart of the Ethereum Blockchain is the Ethereum Virtual Machine, "which can execute code of arbitrary algorithmic complexity. In computer science terms, Ethereum is 'Turing complete'. Developers can create applications that run on the EVM using friendly programming languages modelled on existing languages like JavaScript and Python".
In addition, each and every node "of the network runs the EVM and executes the same instructions".

Finally, note that this gigantic level of "parallelisation of computing across the entire Ethereum network is not done to make computation more efficient. In fact, this process makes computation on Ethereum far slower and more expensive than on a traditional 'computer'. Rather, every Ethereum node runs the EVM in order to maintain consensus across the Blockchain. Decentralized consensus gives Ethereum extreme levels of fault tolerance, ensures zero downtime, and makes data stored on the Blockchain forever unchangeable and censorship-resistant".

4.2 Ether, Account Types, Gas and Transactions

Ether is the name of the currency used within Ethereum. It "is used to pay for computation within the EVM. This is done indirectly by purchasing gas for ether as explained in gas" (Ethereum Foundation 2018b).

Ethereum has a metric system of denominations used as units of Ether. Each denomination has its own unique name. "The smallest denomination aka base unit of ether is called Wei" (Ethereum Foundation 2018b). In Table 27, all denominations are listed.

Table 27: Ether denominations, from the **Ethereum Docs**.

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mewi (lovelace)	1e6 wei	1,000,000
Gewi (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

ERC 20 Tokens

ERC20 is an official protocol where improvements to the Ethereum Blockchain are proposed. The '20' is the unique proposal ID number.

ERC20 defines "a set of rules which need to be met in order for a token to be accepted and called an 'ERC20 Token'. The standard rules apply to all ERC20 tokens since these rules are required to interact with each other on the Ethereum network. These tokens are Blockchain assets that can have value and can be sent and received, like Bitcoin, Litecoin, Ethereum, or any other cryptocurrency.

The difference between these tokens and a standalone currency like Litecoin is that ERC20 tokens piggyback on the Ethereum network, hosted by Ethereum addresses and sent using Ethereum transactions" (Exodus 2018).

Account Types, Gas and Transactions

In Ethereum, every account is represented by an address, an up to 40 characters hexadecimal value. Example: `0×1ae42D3054b8079Ff0694E7B43c230394aFc2125`.

In addition, there are only two types of accounts:

1. **Externally Owned Accounts (EOAs)**: According to the Ethereum Docs[1], an externally controlled account has the following properties:
   - Must have an account balance
   - Can send transactions (Ether transfer or trigger contract code)
   - Is controlled by private keys
   - Has no associated code
2. **Contract Accounts**: According to the same source as EOAs, contract accounts are essentially a smart contract. Properties include:
   - Must have an account balance
   - Must have associated code
   - Code execution is triggered by transactions or messages (calls) received from other contracts
   - When executed, it can perform operations of arbitrary complexity (Turing completeness) as well as manipulate its own persistent storage in addition to the ability to call other contracts

   Externally Owned Accounts are responsible for initiating all transactions in the Ethereum Blockchain. "Every time a contract account receives a transaction, its code is executed as instructed by the input parameters sent as part of the transaction. The contract code is executed by the Ethereum Virtual Machine on each node participating in the network as part of their verification of new blocks" (Ethereum Foundation 2018a).

Transactions

The Ethereum Blockchain breathes transactions, validated by miners every few seconds. One could raise the question: what exactly are transactions? The Ethereum Docs has a fitting explanation:

"The term 'transaction' is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account to another account on the Blockchain".

Transactions contain (Ethereum Foundation 2018a):

- The recipient of the message
- A signature identifying the sender and proving their intention to send the message via the Blockchain to the recipient
- **Value** field — the amount of wei to transfer from the sender to the recipient
- An optional data field, which can contain the message sent to a contract

[1] http://www.ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html

- A **start gas** value, representing the maximum number of computational steps the transaction execution is allowed to take
- A **gas price** value, representing the fee the sender is willing to pay for gas. One unit of gas corresponds to the execution of one atomic instruction, i.e., a computational step.

Gas

When a contract is executed in the Blockchain, every instruction is executed across all nodes in the network. To prevent abuse from bad actors and to reward miners, there is a specified cost associated with executing them, expressed in a number of gas units — alike driving cars: a car owner has to pay for its gas, which has a fixed **gas price**.

The Ethereum Docs have a more in-depth definition of gas:

"Gas is the name for the execution fee that senders of transactions need to pay for every operation made on an Ethereum Blockchain. The name gas is inspired by the view that this fee acts as cryptofuel, driving the motion of smart contracts. Gas is purchased for ether from the miners that execute the code. Gas and ether are decoupled deliberately since units of gas align with computation units having a natural cost, while the price of ether generally fluctuates as a result of market forces. The two are mediated by a free market: the price of gas is actually decided by the miners, who can refuse to process a transaction with a lower gas price than their minimum limit. To get gas you simply need to add ether to your account. The Ethereum clients automatically purchase gas for your ether in the amount you specify as your maximum expenditure for the transaction.

The Ethereum protocol charges a fee per computational step that is executed in a contract or transaction to prevent deliberate attacks and abuse on the Ethereum network. Every transaction is required to include a gas limit and a fee that it is willing to pay per gas. Miners have the choice of including the transaction and collecting the fee or not. If the total amount of gas used by the computational steps spawned by the transaction, including the original message and any sub-messages that may be triggered, is less than or equal to the gas limit, then the transaction is processed. If the total gas exceeds the gas limit, then all changes are reverted, except that the transaction is still valid and the fee can still be collected by the miner. All excess gas not used by the transaction execution is reimbursed to the sender as Ether. You do not need to worry about overspending, since you are only charged for the gas you consume. This means that it is useful as well as safe to send transactions with a gas limit well above the estimates" (Ethereum Foundation 2018a).

Estimating Transaction Costs

Estimating transaction costs is a crucial component of any smart contact or DApp development, since every instruction has an associated cost. For this reason, determining how the gas price is calculated is very important. This can be calculated by two factors (Ethereum Foundation 2018a):

- **Gas used**: total gas consumed by the transaction
- **Gas price**: price, in ether, of exactly one unit of gas specified in the transaction

Total cost, can therefore be calculated: **Total cost** = **gas used** $*$ **gas price**

4.3 Mining, Consensus Algorithms and Blockchain Hardforks

Mining in a Blockchain in short is the process of collecting, validating transactions (in any given cryptocurrency) and assembling these in a "candidate block" with the objective of earning newly created cryptocurrency. A miner is a person/entity executing the process of mining through software. Anyone can perform such task, as cryptocurrency mining is permissionless. This is a very expensive process energy and financially wise.

Mining is linked to consensus algorithms which ensure that when a "candidate block" is mined, the whole Blockchain accepts or rejects this block. If successful, the block is appended to the end of the Blockchain and everyone with a copy of a Blockchain will download this newly added block. There are two main consensus algorithms:

- **Proof of Work:** "A proof of work is a piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements. Producing a proof of work can be a random process with low probability so that a lot of trial and error is required on average before a valid proof of work is generated" (Bitcoin Wiki 2018b).
  The main issue with proof of work is the substantial amount of energy it is required for it to operate
- **Proof of Stake:** "Proof of stake is a different way to validate transactions based and achieve the distributed consensus.
  It is still an algorithm, and the purpose is the same of the proof of work, but the process to reach the goal is quite different.
  Unlike the proof-of-Work, where the algorithm rewards miners who solve mathematical problems with the goal of validating transactions and creating new blocks, with the proof of stake, the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake" (Rosic 2017a)

Figure 13 illustrates how do both consensus algorithms work in addition to their differences.

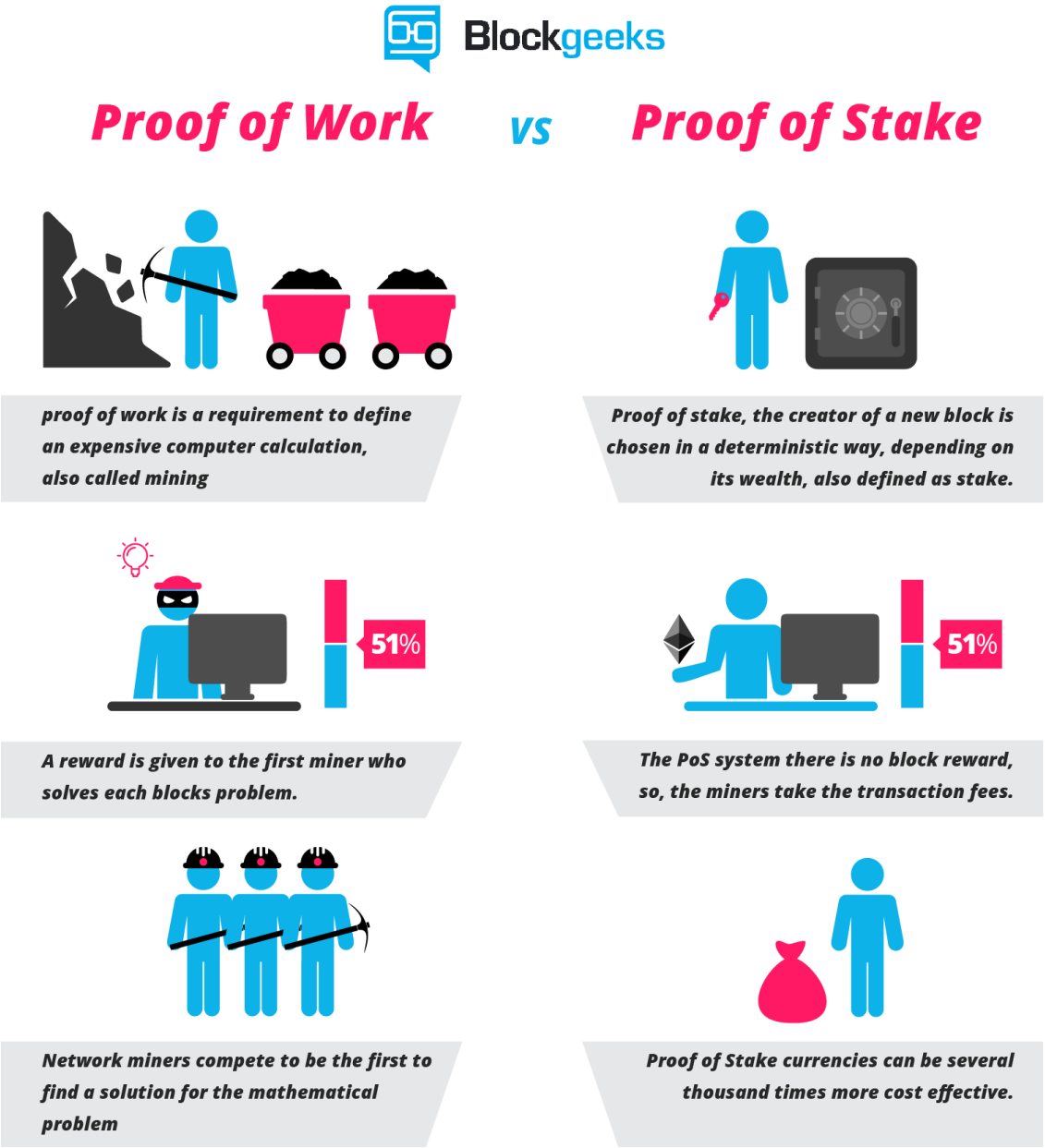

Figure 13: Proof of Work VS Proof of Stake consensus algorithms from Block Geeks.

Finally, the notion of Blockchain hardforks. Every Blockchain, whether the Bitcoin Blockchain, Ethereum or another, has its own set of rules such as block size, transaction time and more. Blockchain hardforks are essentially when a situation occurs and the Blockchain separates into two chains temporarily or permanently due to disagreement on a Blockchain's enforced rules; a necessity to forcibly change these rules or as a result of a nasty hack, such as when the DAO was hacked and a hardfork was needed to preserve the stolen cryptocurrency[2]. Some people debated that hardforking is against the principles of a Blockchain, such as

[2] https://blockgeeks.com/guides/what-is-ethereum-classic/

being immutable[3]. For this reason, Ethereum Classic was created with the stolen cryptocurrency gone, and the Ethereum Blockchain in a state previous to the event.

4.4 Ethereum Clients and Networks

Ethereum clients refer to any Ethereum nodes that can parse and verify the Blockchain, including its smart contracts and beyond. There are several implementations, as seen in Table 28.

Table 28: Ethereum clients list, from the **Ethereum Docs**.

Client	Language	Developers
go-ethereum	Go	Ethereum Foundation
Parity	Rust	Ethcore
cpp-ethereum	C++	Ethereum Foundation
pyethapp	Python	Ethereum Foundation
Ethereum(J)	Java	<ether.camp>
ruby-ethereum	Ruby	Jan Xie
ethereumH	Haskell	BlockApps
ethereumjs-lib	Javascript	Ethereum Foundation

While interacting with an Ethereum node, it is required a JSON-RPC protocol. According to the Ethereum Foundation (2018e), "JSON-RPC is a stateless, Remote Procedure Call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over the HyperText Transfer Protocol (HTTP), or in many various message passing environments. It uses JavaScript Object Notation (JSON) (RFC 4627) as data format".

Moreover, there are some important libraries that implement the JSON-RPC protocol which facilitate the communication between an Ethereum node and an application. For the sake of this dissertation, since Truffle uses JavaScript, **ONLY** the Web3.js API is considered.

Ethereum Networks

In Ethereum, there are two types of networks: **test networks (tesnets)** and the **main Ethereum network**. The latter is the most important one, where DApps are usually deployed. However, when testing smart contracts and DApps in general, a testnet is often the recommended choice. There are three crucial testnets:

1. **Ropsten**[4]
2. **Kovan**[5]

[3] https://cryptovest.com/education/ethereum-classic-explained/
[4] https://github.com/ethereum/ropsten
[5] https://github.com/kovan-testnet/proposal

3. **Rinkeby**[6]

The major difference between testnets and the main network is that on the main network, every transaction has to be paid with real Ether. When developing and testing smart contracts, this becomes unfeasible quickly. Testnets, on the other hand, allow Ether to be generated free of charge, for that specific testnet. In this work, the Rinkeby testnet is used.

Data Protection and Privacy in the Ethereum Blockchain

Data protection and its subset data privacy, is a very important albeit complex and complicated aspect in a person's life. From corporations exploiting user data to maximize profits or simply to leverage users to continue attached to a specific platform, examples of data monopolies still exist today. In this part, the multiple existing solutions that provide data protection and privacy in the Ethereum Blockchain are explored, in addition to some other key aspects such as the different Blockchain types.

Understanding the concept of data privacy and its complexity is essential in any software application nowadays. The Ethereum Blockchain, due to its public design and nature, provides very few options to enforce data privacy in some form between parties. Here, two possible techniques are explored:

- **Zero-knowledge Proofs**: Zero-knowledge proofs allow a user to construct a "mathematical proof that a given program, when executed on some (possibly hidden) input known by the user, has a particular (publicly known) output, without revealing any other information. There are many specialized types of zero-knowledge proofs that are fairly easy to implement; for example, you can think of a digital signature as a kind of zero-knowledge proof showing that you know the value of a private key which, when processed using a standard algorithm, can be converted into a particular public key" (Buterin 2016).

  As for applications of zero-knowledge proofs, these are (Buterin 2016):

  1. **Identity Systems:** "For example, suppose that you want to prove to a system that you are (i) a citizen of a given country, and (ii) over 19 years old. Suppose that your government is technologically progressive, and issues cryptographically signed digital passports, which include a person's name and date of birth as well as a private and public key. You would construct a function which takes a digital passport and a signature signed by the private key in the passport as input, and outputs 1 if both (i) the date of birth is before 1996, (ii) the passport was signed with the government's public key, and (iii) the signature is correct, and outputs 0 otherwise. You would then make a zero-knowledge proof showing that you have an input that, when passed through this function, returns 1, and sign the proof with another private key that you want to use for your future interactions with this service. The service would verify the proof, and if the proof is correct it would accept messages signed with your private key as valid".
  2. **Digital Token Ownership:** in a functioning digital token system, "you do not strictly need to have visible accounts and balances; in fact, all that you need

[6] https://www.rinkeby.io/#stats

is a way to solve the 'double spending' problem — if you have 100 units of an asset, you should be able to spend those 100 units once, but not twice. With zero-knowledge proofs, we can of course do this; the claim that you would zero-knowledge-prove is something like 'I know a secret number behind one of the accounts in this set of accounts that have been created, and it does not match any of the secret numbers that have already been revealed'. Accounts in this scheme become one-time-use: an 'account' is created every time assets are sent, and the sender account is completely consumed. If you do not want to completely consume a given account, then you must simply create two accounts, one controlled by the recipient and the other with the remaining 'change' controlled by the sender themselves".

3. **Two-party Smart Contracts (a financial derivative contract negotiated between two parties):** "When the contract is first negotiated, instead of creating a smart contract containing the actual formula by which the funds will eventually be released (eg. in a binary option, the formula would be 'if index I as released by some data source is greater than X, send everything to A, otherwise send everything to B'), create a contract containing the hash of the formula. When the contract is to be closed, either party can themselves compute the amount that A and B should receive, and provide the result alongside a zero-knowledge-proof that a formula with the correct hash provides that result. The Blockchain finds out how much A and B each put in, and how much they get out, but not why they put in or get out that amount". Figure 14 illustrates this.

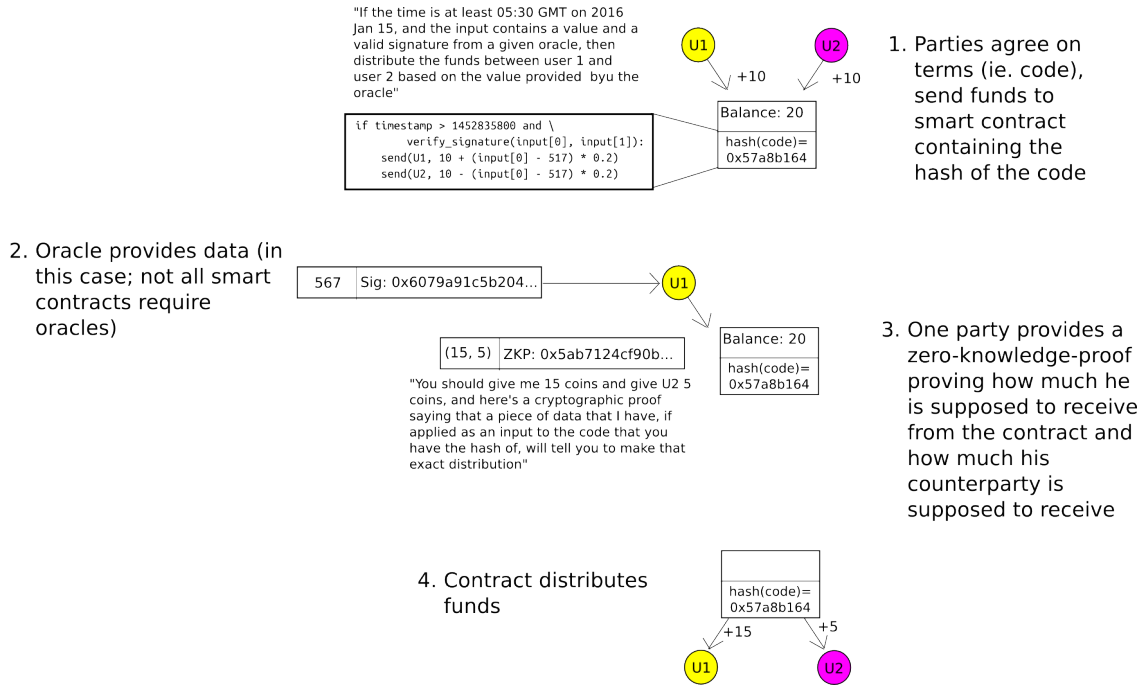

Figure 14: Application of Zero-knowledge Proofs (Buterin 2016).

- **Ring Signatures**: A technology which is moderately and technically complicated, albeit "extremely promising for both token anonymization and identity applications, is ring signatures. A ring signature is essentially a signature that proves that the signer has a private key corresponding to one of a specific set of public keys, without revealing

> which one. The two-sentence explanation for how this works mathematically is that a ring signature algorithm includes a mathematical function which can be computed normally with just a public key, but where knowing the private key allows you to add a seed to the input to make the output be whatever specific value you want. The signature itself consists of a list of values, where each value is set to the function applied to the previous value (plus some seed); producing a valid signature requires using knowledge of a private key to 'close the loop', forcing the last value that you compute to equal the first. Given a valid 'ring' produced in this way, anyone can verify that it is indeed a 'ring', so each value is equal to the function computed on the previous value plus the given seed, but there is no way to tell at which 'link' in the ring a private key was used" (Buterin 2016).

In addition to the privacy techniques mentioned above, there are more robust and complex solutions to provide in some form data protection and privacy to end users. These are:

1. **Quorum**: "A permissioned implementation of Ethereum supporting data privacy"[7]
2. **Hawk**: "Privacy-Perserving Blockchain and Smart Contracts"[8]
3. **CoinJoin**: "Bitcoin privacy for the real world"[9]

Quorum

Quorum is a private or permissioned Blockchain based on the "official Go implementation of the Ethereum protocol[10]. Quorum uses a voting based consensus algorithm, and achieves data privacy through the introduction of a new 'private' transaction identifier. One of the design goals of Quorum is to reuse as much existing technology as possible, minimizing the changes required to go-ethereum in order to reduce the effort required to keep in sync with future versions of the public Ethereum code base. Much of the logic responsible for the additional privacy functionality resides in a layer that sits atop the standard Ethereum protocol layer. Figure 15 below provides a high-level overview of the Quorum Blockchain platform and architectural components" (J.P. Morgan 2016).

[7] https://github.com/jpmorganchase/quorum
[8] http://oblivm.com/hawk/index.html
[9] https://bitcointalk.org/index.php?topic=279249.0
[10] https://github.com/ethereum/go-ethereum

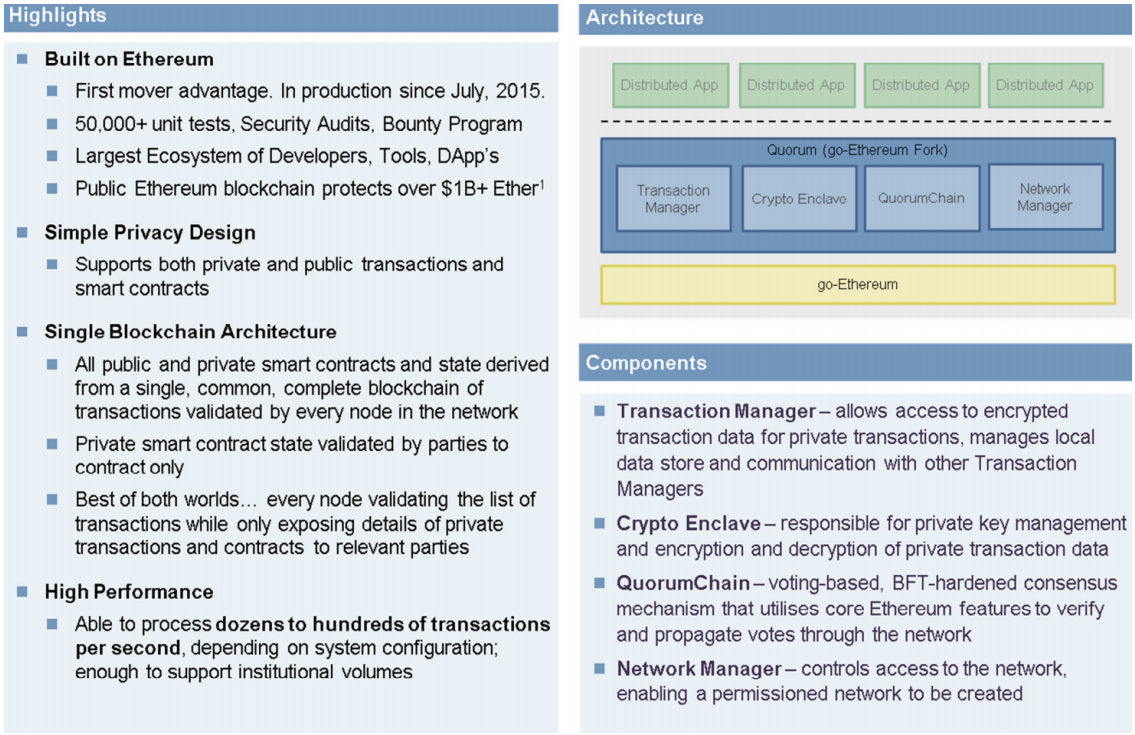

Figure 15: Quorum overview (J.P. Morgan 2016).

The essence of Quorum is to use cryptography to prevent all except those party to the transaction from seeing sensitive data. "The solution involves a single shared Blockchain and a combination of smart contract software architecture and modifications to Ethereum. The smart contract architecture provides segmentation of private data. Modifications to the go-ethereum codebase include modifications to the block proposal and validation processes. The block validation process is modified such that all nodes validate public transactions and any private transactions they are party to by executing the contract code associated with the transactions. For other 'private transactions', a node will simply skip the contract code execution process.

This will result in a segmentation of the state database, i.e. the state database is split into a private state database and a public state database. All nodes in the network are in perfect state consensus on their public state. The private state databases will differ. Even though the client node state database no longer stores the state of the entire global state database, the actual distributed Blockchain and all the transactions therein are fully replicated across all nodes and cryptographically secured for immutability.
This is an important distinction relative to other segmentation strategies based on multiple block chains and adds to the security and resiliency of the design — Figure 16" (J.P. Morgan 2016).

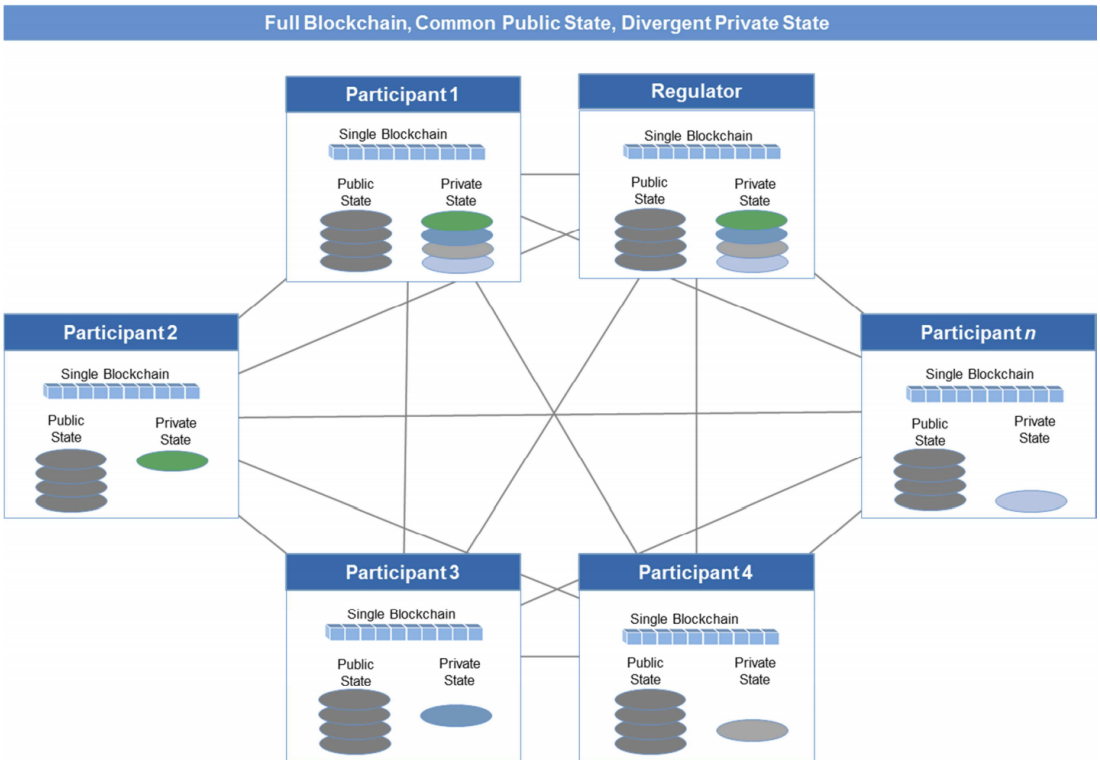

Figure 16: Quorum network (J.P. Morgan 2016).

"Quorum uses a majority voting protocol dubbed QuorumChain, where a subset of nodes within the network are given the ability to vote on blocks. The voting role allows a node to vote on which block should be the canonical head at a particular height. The block with the most votes will win and is considered the canonical head of the chain.

Data privacy in Quorum is achieved through cryptography and segmentation. Cryptograhy is applied to the data in transactions, which everyone sees on the Blockchain. Segmentation is applied to each node's local state database which contains the contract storage and is only accessible to the node. Only nodes party to private transactions are able to execute the private contract code associated with the transactions which results in updating the private contract data storage in the local state database. The result is that each node's local state database is only populated with public and private data they are party to. Figure 17 below outlines the high-level logical design of the Quorum privacy solution" (J.P. Morgan 2016).

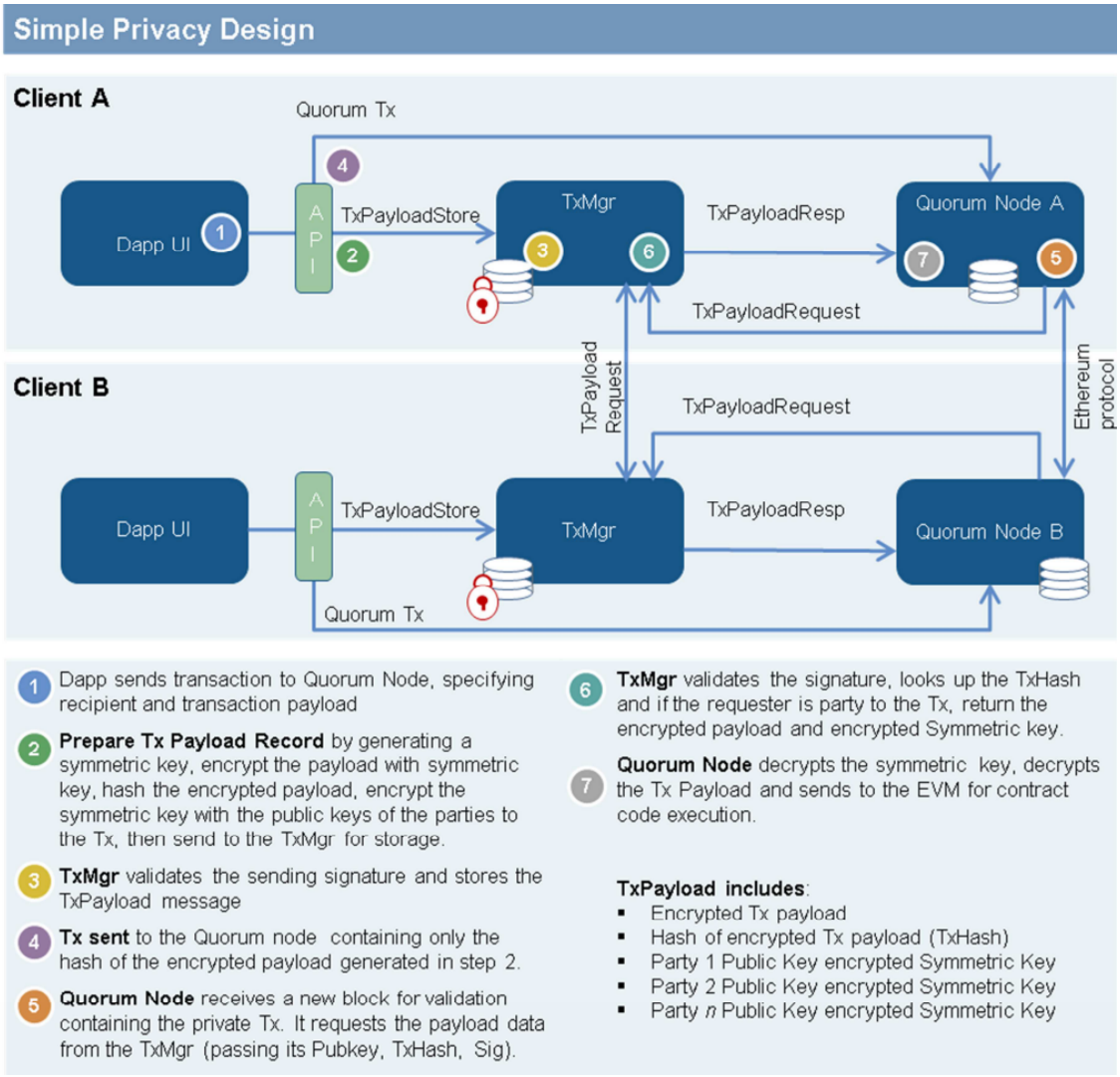

Figure 17: Simple privacy design (J.P. Morgan 2016).

Quorum facilities data privacy in terms of:

1. "**Private Transactions:** Quorum Transactions take a list of public keys that identify the parties to the transaction and therefore make the transaction private to those parties. With this list, a standard Ethereum transaction is generated where the payload is simply the hash of the encrypted private data. This newly formed transaction carrying only the cryptographic hash is sent to the Quorum node where it is distributed to all the nodes in the network as a pending transaction.
   A Quorum transaction contains:
   - The recipient
   - Signature identifying the sender
   - An amount of ether (although having an ether balance is not required within Quorum)
   - An optional list of parties that the transaction should be private to

- An optional data field (containing a hash in the case of a private transaction)

Figure 3 illustrates the basic steps in producing and sending private transactions.

2. **Private Contracts:** A private contract is a contract that was created by a private transaction" (J.P. Morgan 2016)

Hawk

Hawk is a Blockchain-based smart contract system that "stores encrypted transactions on the Blockchain, and relies on cryptography to retain the security of the cryptocurrency.
Hawk is a Blockchain-based smart contract system that stores encrypted transactions on the Blockchain, and relies on cryptography to retain the security of the cryptocurrency" (Kosba et al. 2016).
A Hawk program contains two parts (see Figure 18).

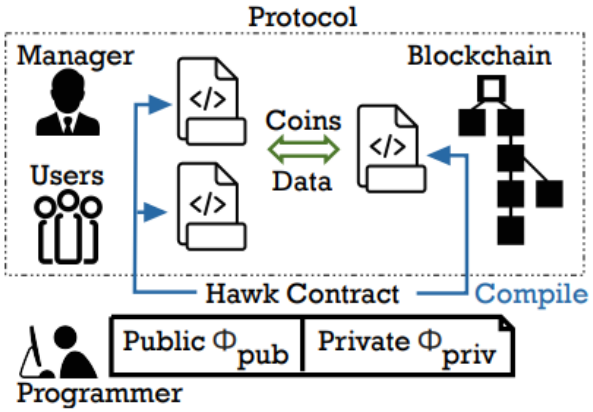

Figure 18: Hawk overview (Kosba et al. 2015).

1. "A private portion denoted ϕ_{priv} which takes in parties' input data (e.g., choices in a 'rock, paper, scissors' game) as well as currency units (e.g., bids in an auction). ϕ_{priv} performs computation to determine the payout distribution amongst the parties. For example, in an auction, winner's bid goes to the seller, and others' bids are refunded. The private Hawk program ϕ_{priv} is meant to protect the participants' data and the exchange of money
2. A public portion denoted ϕ_{pub} that does not touch private data or money"(Kosba et al. 2015)

Hawk provides security guarantees (Kosba et al. 2015):

1. "**On-chain Privacy**: stipulates that transactional privacy be provided against the public (i.e., against any party not involved in the contract) — unless the contractual parties themselves voluntarily disclose information. Although in Hawk protocols, users exchange data with the Blockchain, and rely on it to ensure fairness against aborts, the flow of money and amount transacted in the private Hawk program ϕ_{priv} is cryptographically hidden from the public's view. Informally, this is achieved by sending 'encrypted' information to the Blockchain, and relying on zero-knowledge proofs to enforce the correctness of contract execution and money conservation

2. **Contractual Security**: while on-chain privacy protects contractual parties' privacy against the public (i.e., parties not involved in the financial contract), contractual security protects parties in the same contractual agreement from each other. Hawk assumes that contractual parties act selfishly to maximize their own financial interest. In particular, they can arbitrarily deviate from the prescribed protocol or even abort prematurely. Therefore, contractual security is a multi-faceted notion that encompasses not only cryptographic notions of confidentiality and authenticity, but also financial fairness in the presence of cheating and aborting behavior".

Hawk also guarantees the following contractual security requirements for parties in the smart contract (Kosba et al. 2015):

- "**Input Independent Privacy**: each user does not see others' bids before committing to their own (even when they collude with a potentially malicious manager). This way, users bids are independent of others' bids
- **Posterior Privacy**: as long as the manager does not disclose information, users' bids are kept private from each other (and from the public) even after the auction
- **Financial Fairness**: parties may attempt to prematurely abort from the protocol to avoid payment or affect the redistribution of wealth. If a party aborts or the auction manager aborts, the aborting party will be financially penalized while the remaining parties receive compensation. Hawk also allows the programmer to define additional rules, as part of the Hawk contract, that govern financial fairness"

CoinJoin

CoinJoin is a protocol that facilitates privacy. See Figure 19 below to understand how it works.

1. "N parties come together over some anonymous channel, eg. Tor. They each provide a destination address $D[1] \ldots D[N]$
2. One of the parties creates a transaction which sends one coin to each destination address
3. The N parties log out and then separately log in to the channel, and each contribute one coin to the account that the funds will be paid out from
4. If N coins are paid into the account, they are distributed to the destination addresses, otherwise they are refunded

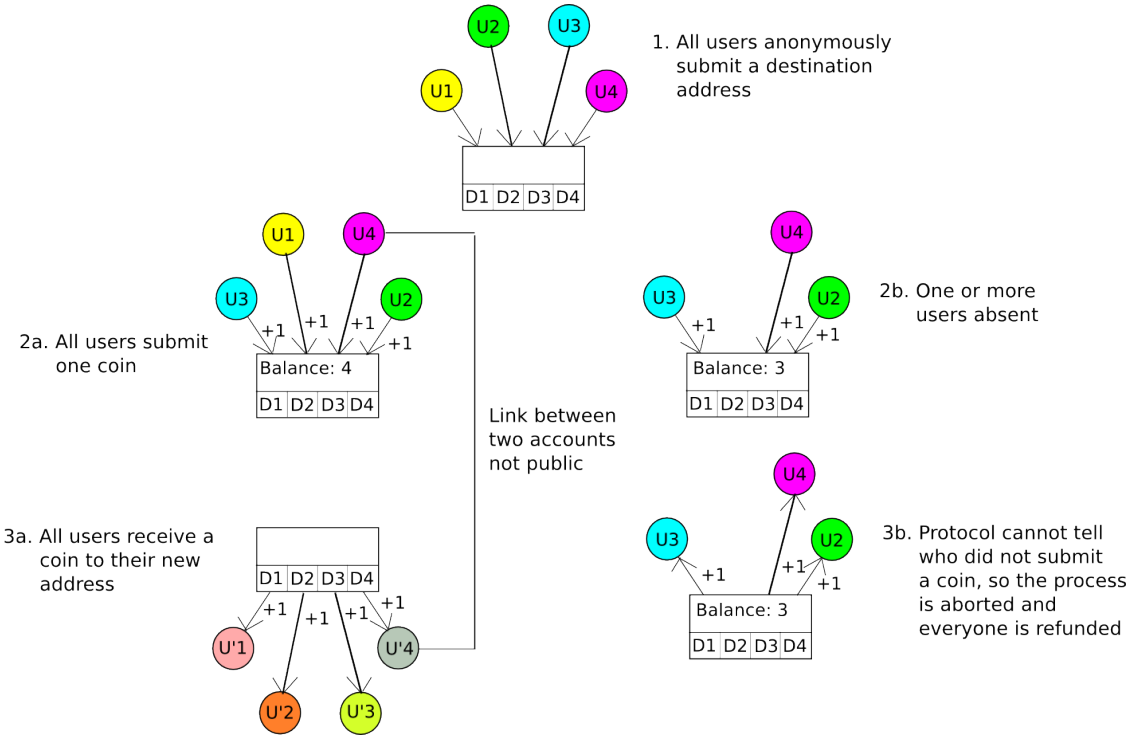

Figure 19: CoinJoin example (Buterin 2016).

If all participants are honest and provide one coin, then everyone will put one coin in and get one coin out, but no one will know which input maps to which output. If at least one participant does not put one coin in, then the process will fail, the coins will get refunded, and all of the participants can try again" (Buterin 2016).

Private/Federated and Public Blockchains

There are two types of Blockchains: a private, permissioned Blockchain and a public one. Public Blockchains are the most common factor, since they use Distributed Ledger Technology to process, record and validate transactions. However, there is also a need for private ones: "private institutions like banks realized that they could use the core idea of Blockchain as a Distributed Ledger Technology, and create a permissioned Blockchain (private or federated), where the validator is a member of a consortium or separate legal entities of the same organization.

Private Blockchains are valuable for solving efficiency, security and fraud problems within traditional financial institutions, but only incrementally. It is not very likely that private Blockchains will revolutionize the financial system. Public Blockchains, however, hold the potential to replace most functions of traditional financial institutions with software, fundamentally reshaping the way the financial system works" (BlockchainHub 2018). Table 29 illustrates the differences between both types of Blockchains.

Table 29: Public VS private Blockchains from Chris Skinner's Blog.

	Public	Private/Federated
Access	• Open read/write	• Permissioned read and/or write
Speed	• Slower	• Faster
Security	• Proof of Work • Proof of Stake • Other Consensus Mechanisms	• Pre-approved participants
Identity	• Anonymous • Pseudonymous	• Known identities
Asset	• Native Asset	• Any asset

In addition, there is also the notion of permissioned VS permissionless Blockchains. Figures 20 and 21 show a comparison between these two types.

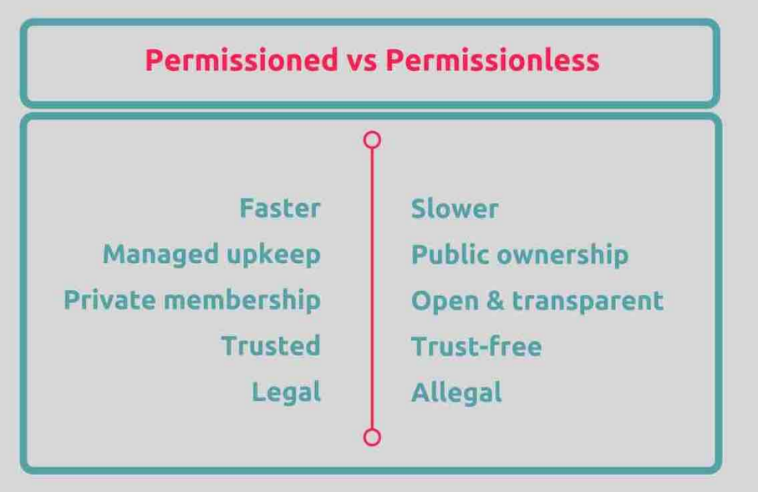

Figure 20: Permissioned vs permissionless Blockchains from Calvin Wood.

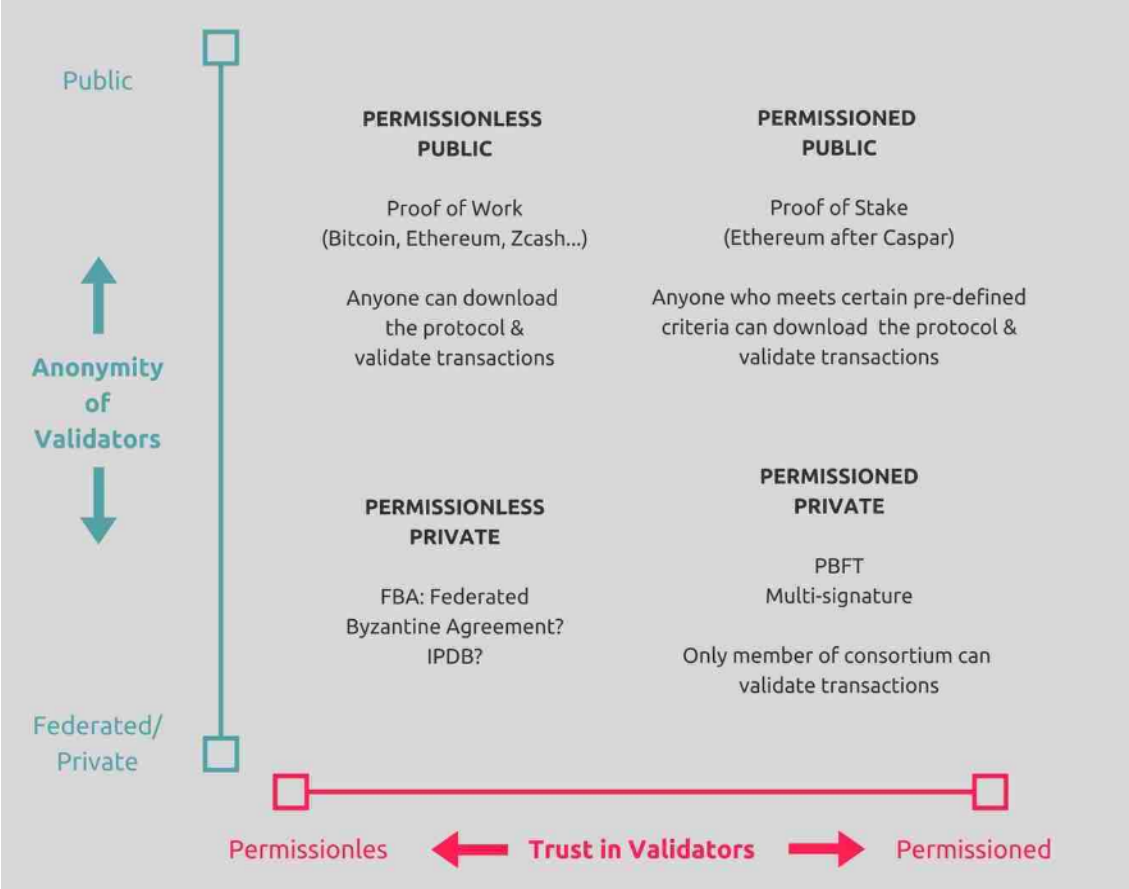

Figure 21: Permissioned vs permissionless Blockchains part 2 from BlockchainHub.

Chapter 5

Case Study

This chapter details the technologies (that have not been explained before) and the requirements of this dissertation's case study: the DFiles Decentralized Application. It is open-source and developed with decentralized technologies such as the IPFS and Ethereum smart contracts with a centralized component for user authentication. Further, DFiles also aims to adhere to Blockchain Software Engineering (Destefanis et al. 2018) by going through each software development phase, explained starting this chapter:

- Requirement gathering and analysis
- Design
- Implementation
- Testing
- Deployment

5.1 Technologies

In addition to Ethereum and its smart contracts, the DFiles DApp was built with many other technologies, presented in two categories:

- **Backend**: developed with NodeJS/Express, the IPFS and the document database MongoDB:
  - **NodeJS** is a free, open-source server environment which uses JavaScript and runs on multiple platforms[1]
  - **ExpressJS** is a "minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications"[2]
  - **IPFS** is a "peer-to-peer hypermedia protocol to make the web faster, safer, and more open"[3]. When developing Decentralized Applications, storing data in the Blockchain is not feasible at the moment due to its substantially high gas cost. To solve this, IPFS allows to address large amounts of data by storing IPFS links in each Blockchain transaction. This way, data stored is secure.

[1] https://www.w3schools.com/nodejs/nodejs_intro.asp
[2] https://expressjs.com/
[3] https://ipfs.io/

However, as it will later be discussed in section 9.3, limitations and future work, the IPFS only stores files that each node is interested in

- **MongoDB** is a document database with significant scalability and flexibility[4].

  It has the following features[5]:

  * MongoDB stores data in flexible, JSON-like documents
  * As a distributed database at its core, MongoDB provides availability, horizontal scaling and geographic distribution

- **Frontend**: built with Bootstrap, Flexbox, SASS and ReactJS:
  - **Bootstrap** is an "open source toolkit for developing with Hypertext Markup Language (HTML), Cascade Style Sheets (CSS), and JavaScript"[6]. Moreover, it allows in a simple way, the development of Web applications for multiple devices: **one design, multiple devices**. This concept is called **responsive design** — the web application adapts to the screen size of the user's device. Bootstrap has five main screen size categories[7]:
    * **Extra Small**: devices with screen widths less than 576px (very small phones)
    * **Small**: devices with screen widths less or equal to 576px (landscape phones)
    * **Medium**: devices with screen widths less or equal to 768px (tablets)
    * **Large**: devices with screen widths less or equal to 992px (laptops)
    * **Extra Large**: devices with screen widths less or equal to 1200px (large desktop computers)

    Its biggest strength is its impeccable grid system. Table 30 illustrates this feature.

Table 30: Bootstrap grid layout, from Bootstrap docs.

	Extra small	**Small**	**Medium**	**Large**	**Extra Large**
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
Number of columns	12				
Gutter width	30px				
Nestable	Yes				
Column ordering	Yes				

In bootstrap, each screen row has 12 columns. Depending on screen size, the columns can be:

[4] https://www.mongodb.com/what-is-mongodb
[5] https://www.mongodb.com/what-is-mongodb
[6] https://getbootstrap.com/
[7] https://getbootstrap.com/docs/4.0/layout/grid/

  - **.col-**: for extra small devices
  - **.col-sm-**: for small devices
  - **.col-md-**: for medium devices
  - **.col-lg-**: for large devices
  - **.col-xl-**: for extra large devices

  With this column system, web applications can be flexible enough to easily adapt to the user's screen size.

- **Flexbox** is a flexible box layout module which makes it easier to design flexible layouts[8].
  Combining it together with Bootstrap when developing Web applications, it allows with little effort to develop them for any device.
- **SASS** is a mature, stable and powerful professional grade CSS extension[9].
  Its biggest improvement is the ability to use variables inside styling files (.scss).
- **ReactJS** is a JavaScript library used for building user interfaces. It has the following features[10]:
  - **Declarative**: React ensures interactive User Interfaces (UIs) are created in a painless way, by designing simple views for each application state. React will render and update efficiently when components change
  - **Component-Based**: components in React are encapsulated and they manage their own application state. By mixing several components, a robust and secure Web application can be built
  - **Learn Once, Write Anywhere**: React can be used in any technology stack, at any time

5.2 Requirements

In standard, centralized software applications, there are several phases to properly develop software applications: the software development lifecycle. In Decentralized Applications, however, there must be special attention to develop secure smart contracts with a special emphasis on testing.

In any software application, defining its multiple requirements is one of the most important phases in software development.
In this section, the multiple requirements are detailed and presented for the DFiles DApp. These are included in two main categories: **functional** and **non-functional** requirements.

[8] https://www.w3schools.com/css/css3_flexbox.asp
[9] https://sass-lang.com/
[10] https://reactjs.org/

Functional requirements, are, by definition, **any requirement which specifies what the system should do**[11].

As for the DFiles DApp, these are:

1. Provide general information about DFiles
2. Register a new user account with hashed passwords
3. The minimal user age required to use DFiles is 16
4. Provide to the user information about the DFiles privacy policy
5. Login to existing user accounts
6. The DApp must be able to communicate with the Ethereum Blockchain via the user's Metamask Google Chrome/Firefox extension
7. Encrypt user files before their upload process to the IPFS
8. Upload user files to the IPFS
9. Register each user uploaded file Universal Resource Locator (URL) in an Ethereum transaction
10. List all the user uploaded files to the IPFS
11. Allow for users to:
    - Delete their account
    - Edit their account
12. Provide users with a copy of their data and the ability to download it

In addition to this, Figure 22 is a use case diagram which also illustrates the functional requirements of DFiles.

[11] https://reqtest.com/requirements-blog/understanding-the-difference-between-functional-and-non-functional-requirements/

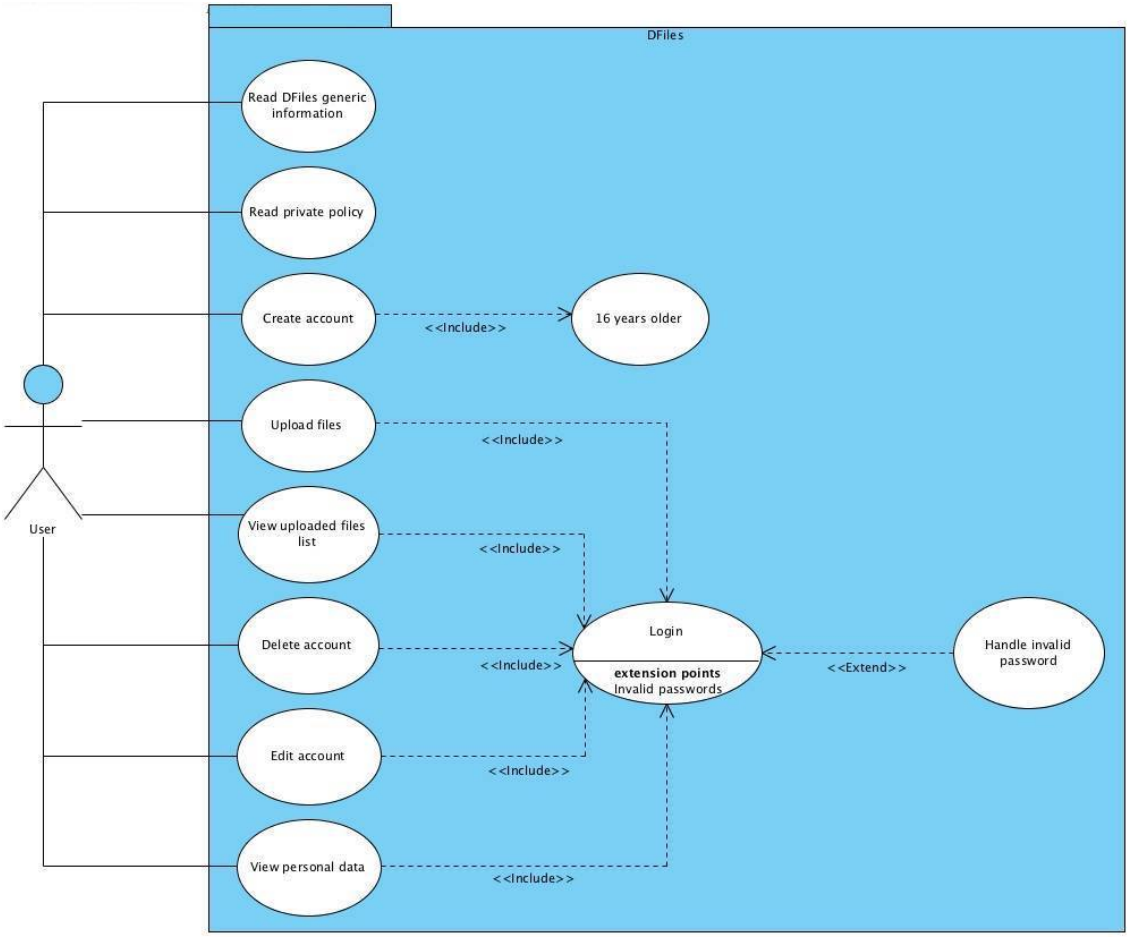

Figure 22: DFiles functional requirements use case diagram.

Non-functional requirements, are, by definition, **any requirement which specifies how the system performs a certain function**[12].

In the DFiles DApp, these include:

- The DFiles homepage should load within 5 seconds
- Password hashing when a user is creating an account should not take longer than 5 seconds
- Storing user accounts in the MongoDB database should not take longer than 20 seconds
- When the user is registering its account, his information must be stored in a secure way
- The DFiles privacy policy loading time should not be longer than 15 seconds
- Fetching user data, while he attempts to login, should not take more than 30 seconds
- File data encryption should be successful and not take more than one minute

[12] https://reqtest.com/requirements-blog/understanding-the-difference-between-functional-and-non-functional-requirements/

- User files uploaded to the IPFS must not take longer than 5 minutes
- User files must have valid extensions:
  - .csv
  - .pdf
  - .docx
  - .xlsx
  - .pptx
  - .txt
  - .png
  - .jpeg
  - .bmp
- Each Ethereum transaction must be successful for the user to be able to upload files to the IPFS
- The list of user files must be displayed within 10 seconds
- Data fetched from the MongoDB database when users request to edit their accounts must be accurate
- Data attached to users must be deleted from the MongoDB within one hour when the user requests for it to be deleted
- Data must be accurately updated when the user edits his account information
- Data must be accurate and displayed within 5 seconds to the user when he requests to edit his personal information
- Provide users with an accurate copy of their data within 30 minutes

Chapter 6

Blockchain Software Engineering

In the previous chapter, the requirements for the DFiles DApp were explained. This is the first phase of Blockchain Software Engineering: software engineering adapted for Blockchain technology such as Decentralized Applications. In this chapter, the remaining phases of BOSE are detailed:

- Design
- Implementation
- Testing
- Deployment

However, note that this chapter **does not** include any discussion or implementation of the General Data Protection Regulation. This is detailed in chapter 7.

6.1 Design and Implementation

Design and implementation are two of the most important stages of BOSE. In DFiles, there are two distinct system architectures: the first one uses a tool that locally simulates the Ethereum Blockchain, Ganache and a local IPFS node. This was the one chosen for almost the entirety of this work, as it was easier to write, test and deploy smart contracts. By using a local IPFS node, it was possible to fully assert, locally, if data encryption is feasible or not (this is discussed in chapter 8, evaluation) in Ethereum Homestead. This system architecture is shown in Figure 23, as an Unified Modeling Language (UML) diagram.

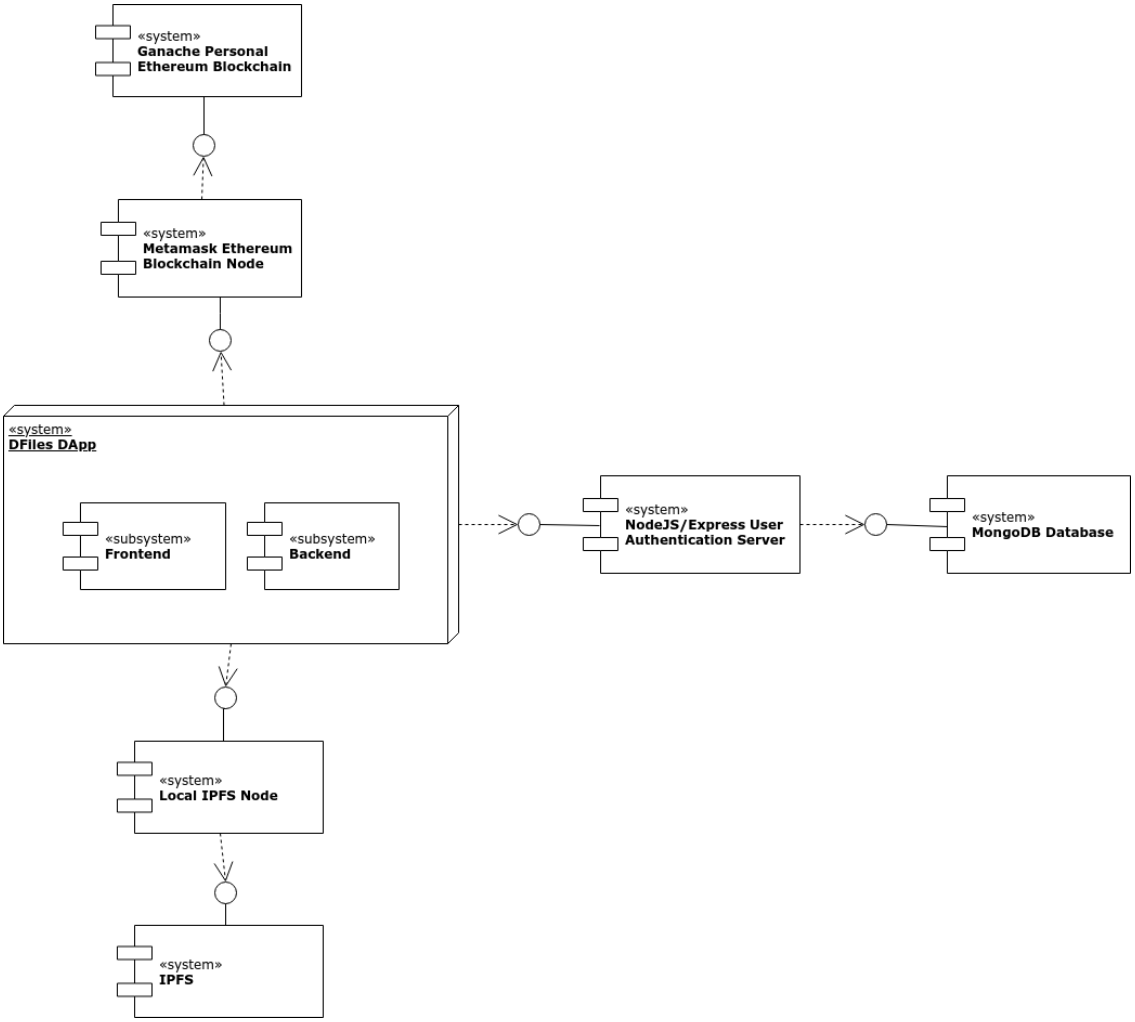

Figure 23: The DFiles local system architecture diagram.

The DFiles DApp has two main components:

- **Frontend**: this is responsible to display information in a clear, user-friendly and simple way to the DFiles user. Here, ReactJS, HTML5, Bootstrap, Flexbox and SASS are used for this purpose
- **Backend**: it interacts with the frontend to fetch business logic from multiple sources. In DFiles, these include:
  - **User Authentication** with NodeJS/Express server and a MongoDB database
  - **Ethereum Smart Contracts** which ensure business logic is enforced until some point in the distant future
  - **IPFS** for decentralized file storage

  As seen in Figure 23, the Implementation of the DFiles DApp is connected to a NodeJS/Express server for simple user authentication which fetches user data from a MongoDB database. Futhermore, the DApp is connected to local IPFS node, which in turn is connected to the IPFS network. Similarly, DFiles is also attached to a Metamask

Ethereum node, which ultimately is connected to private network with Ganache: this simulates, locally, the Ethereum Blockchain.

- **Full stack**: the Truffle[1] framework was used as a bridge between the frontend and the backend, with the Web3JS API[2]

The other system architecture was used in the later stages of this work, after the deployment phase. It is also illustrated in the form of an UML diagram: Figure 24.

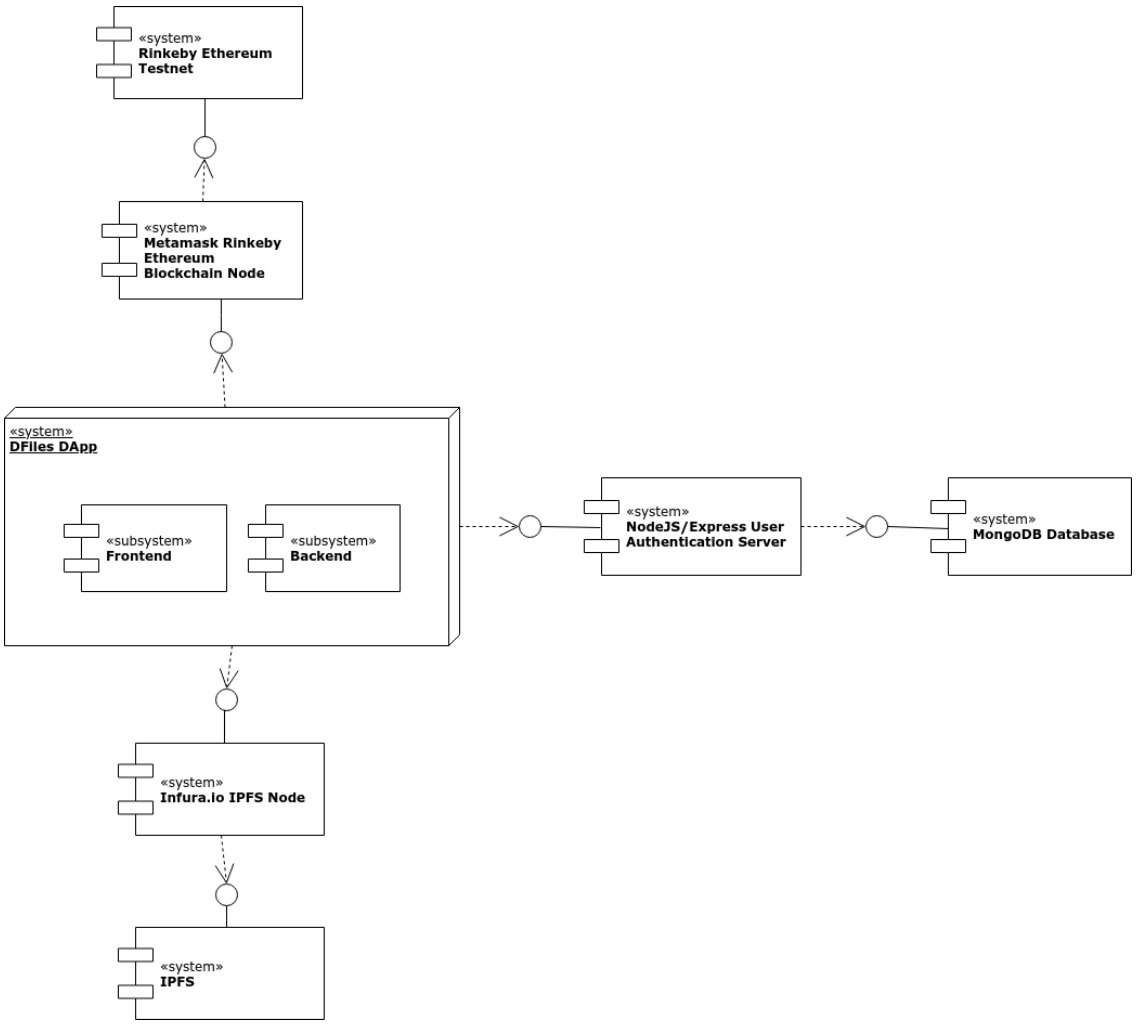

Figure 24: The final DFiles system architecture diagram.

The main difference between the two system architectures is that the final one connects to an Infura.io IPFS node and to the Ethereum Rinkeby test network via metamask. It does not, therefore, use a local IPFS node and Ganache. The rest remains the same.

[1] https://truffleframework.com/
[2] https://github.com/ethereum/wiki/wiki/JavaScript-API

Folder Struture

In addition to specifying DFiles system architecture, the overall folder structure must be explained. In part, this was already defined by the Truffle Framework. In the rest of this chapter, this folder structure will include more detail in the relevant areas (for example, in the frontend, the folder structure is explained thoroughly concerning **only** the frontend). For now, a more generic approach is taken for explaining the overall file structure.

Truffle, by default, organizes files and folders in a certain way:

- **build**: compiled smart contracts are stored here, in JSON files
- **contracts**: Solidity smart contracts (files ending in .sol) must be written here
- **migrations**: migrations are JavaScript files that help one to deploy "contracts to the Ethereum network. These files are responsible for staging deployment tasks, and they are written under the assumption that the deployment needs will change over time. A history of previously run migrations is recorded on-chain through a special Migrations contract"[3]
- **public**: files and folders that can be accessed publicly. Here, the **index.html** file is included, which is important for the ReactJS frontend section
- **scripts**: JavaScript files that are responsible for starting, building and testing the DFiles DApp
- **src**: all the source files for the frontend part of the DApp are stored here
- **test**: all the JavaScript files that use Mocha[4] and Chai[5] for smart contract unit testing

6.1.1 Backend

In this subsection, the backend of the DFiles DApp is explained:

- **Centralized:**
  - The NodeJS/Express overall server structure with its routes
  - The MongoDB schema for user authentication
- **Decentralized:**
  - The core smart contracts are displayed here, in addition to a strong emphasis on security with static analysis tools such as Securify and Smart Check and Solidity design patterns

NodeJS/Express and MongoDB

In the DFiles DApp, there must be some form of user authentication. For this reason, NodeJS/Express server was chosen for this task with an Application Programming Interface

[3] https://truffleframework.com/docs/truffle/getting-started/running-migrations
[4] https://mochajs.org/
[5] http://www.chaijs.com/

(API) applying the Representational State Transfer (REST) architectural style.
This API also uses the Model View Controller (MVC) architectural pattern: it allows the separation of an application into three core parts[6]:

- **Model**: it represents all the business logic-data. Typically, models are created similar to classes in Object Oriented Programming (OOP) with their own set of rules. For example, in a user registration API, the model could be called "User" with all its own properties such as email, address or phone number
- **Controller**: it acts as an interface between the Model and the View components. Its functions are to process incoming requests and core business logic, manipulate data from the Model component and finally dispatch data to the Views component in order for it to render the final output
- **View**: its only function is to render User Interface logic coming from the Controller component. In other words, this component receives business logic from the Controller component, and is responsible to display it to the user

In terms of overall folder structure, most folders were created to obey to the MVC pattern:

- **app** folder:
  - **controllers**: this folder contains the MVC controllers; it only has one file for handling user logic when registering/logging in. This file also works as a view, since it returns a HTTP response
  - **models** folder: it contains all the MVC models; it only has the user model
  - **routes** folder: contains the user REST API routes
  - **config**: this folder is responsible for MongoDB connection configuration
  - **Other Files**:
    - **server.js**: main server file

  Figure 25 better illustrates the different files and folders present in the server.

Figure 25: NodeJS/Express server folder structure.

The NodeJS/Express server stores, reads and updates data from a MongoDB database. It uses Mongoose[7] for Object Modelling (OM).
For the centralized authentication system, it has only one schema: the user's schema:

- **Username**: user's username, with the following properties:

[6] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
[7] https://mongoosejs.com/

        - **Type**: string
        - **Index**: true
        - **Unique**: true
        - **Required**: true
        - **Email**: user's email, with the following properties:
            - **Type**: string
            - **Required:** true
        - **Password**: user's password (hashed) with the following properties:
            - **Type**: string
            - **Required**: true
        - **EthereumAccountAddress**: the user's Ethereum Address, which has the following properties:
            - **Type**: string
            - **Required**: true
        - **PrivateKey**: the user's auto generated, unique private key
        - **PublicKey**: the user's auto generated, unique public key
- Ethereum smart contracts
- IPFS

Routes

In the REST API, routes work as endpoints starting from the root, /. There are multiple HTTP methods such as POST and GET, which return successful or error HTTP codes with a message.
These routes are defined in the routes and controllers folders. Table 31 summarizes them. These routes include:

- **GET** /: presents a greeting message
- **POST /users**: expects user username, email, password, ethereum account address and auto generated private key to be sent in the body. Password is hashed and stored in the MongoDB database. If this is successful, the new user is created and saved in the database. Otherwise, HTTP error code 500 is sent with an error message
- **GET /users/:login**: with username and password supplied in the query string (ex: /users/login?username=myUsername&password=myPassword), this route returns HTTP error code 404 if the user or password is invalid, HTTP error code 500 if there was an unknown error. If the user exists and the password is correct, this route returns the username, email address, private key and Ethereum account address
- **GET /users/:username**: route to fetch all details of a specific user from the MongoDB database. With the user supplied as a parameter (ex: /users/johnSmith), it

will return HTTP error code 404 if the username is not found; 500 if there was an unknown error and all details of the specific user are returned otherwise

- **PUT /users/:username**: route to update a user with username supplied as parameter. If this username is found in the MongoDB database, it is updated. If it is not found, HTTP error code 404 is returned. If an unknown error occurs, HTTP error code 500 is returned
- **DELETE /users/:username**: route to delete a user with username supplied as parameter. If the user is found via its username, it is deleted. Otherwise, if an unknown error occurs, HTTP error code 500 is returned; if the user is not found, the HTTP error code 404 is returned instead

Table 31: NodeJS/Express authentication server routes.

Method	Route
GET	/
POST	/users
GET	/users/:login
GET	/users/:username
PUT	/users/:username
DELETE	/users/:username

Ethereum Smart Contracts

Ethereum and its smart contracts is the core technology in this work. Because of this, all aspects relating to Ethereum and its smart contracts have to be thoroughly documented. To do this, several aspects must be taken into account:

- **Design Patterns**: smart contract development with Solidity has evolved substantially in the last couple of months. Akin to traditional software development, design patterns emerged to help solve a significant amount of problems in smart contracts due to its immutability, impossibility to patch bugs and vulnerabilities post deployment. Thus, documenting the possible design patterns for a DApp, is one of the most important tasks when writing smart contracts
- **Security**: another important issue when developing smart contracts. These are immutable, impossible to patch for bugs and vulnerabilities once they are deployed. For this reason, ensuring smart contracts are secure even before deployment is a must for any DApp to be successful
- **Static Analysis**: as the name might imply, static analysis is debugging code before it is run. Together with security vulnerability checking, smart contracts can be secure, fast, and deployed for many years to come

Design Patterns

Certainly one of the most important aspects to consider when developing any software application, is the knowledge and further choice of the right design patterns. In Ethereum smart contracts in particular, these have an extra layer of importance due to its immutability

and impossibility to patch post deployment.
For this reason, the most relevant smart contract design patterns are explained in-depth, each belonging to five different categories (Wöhrer and Zdun 2018):

1. **Action and Control**
2. **Authorization**
3. **Lifecycle**
4. **Maintenance**
5. **Security**

The **state machine** pattern (action and control category) solves the issue of when an application has different behavioral stages and transitions by applying a state machine to model and represent the different transitions and their behavioral contract states (Wöhrer and Zdun 2018).
In Appendix C, there is an example of this pattern being applied, where a state machine is used to represent a deposit lock and therefore accepts deposits for a period of a day and releases them after seven days (Wöhrer and Zdun 2018).

The **ownable** pattern belongs to the authorization category: a group of patterns that control access to smart contract functions and provide basic authorization control.
The problem at hand is that "any party can call a contract method, but it must be ensured that sensitive contract methods can only be executed by the owner of a contract" (Wöhrer and Zdun 2018). The solution is to store the contract creator's address as the owner of a contract.
In Appendix D, there is an example of the application of this pattern.

The **access restriction** pattern belongs to the same category of the Ownable pattern: authorization.
Typically, in a smart contract, a contract method is executed without being checked for preconditions. In this case, it is often desired the its execution is only allowed if certain conditions are met.
The solution to this problem is simple: define and apply applicable modifiers which check for the desired requirements (Wöhrer and Zdun 2018).
Appendix E exemplifies the usage of this pattern.

Mortal belongs to the lifecycle category: "a group of patterns that control the creation and destruction of smart contracts" (Wöhrer and Zdun 2018). The problem to solve here is the issue of the immutability of smart contracts once deployed. Furthermore, when a smart contract is in fact deployed, and its lifetime is over, it must be possible to destroy and stop it from operating.
Solution: "Use a selfdestruct call within a method that does a preliminary authorization check of the invoking party.
A contract is defined by its creator, but the execution, and subsequently the services it offers are provided by the Ethereum network itself. Thus, a contract will exist and be executable as long as the whole network exists, and will only disappear if it was programmed to self destruct. Mortal is a pattern that enables the creator of a contract to destroy it. The

pattern uses a modifier to ensure that only the owner of the contract can execute the selfdestruct operation, which sends the remaining Ether stored within the contract to a designated target address (provided as argument) and then the storage and code is cleared from the current state" (Wöhrer and Zdun 2018). Appendix F exemplifies the application of this pattern.

The **Satellite** pattern belongs to the category of maintenance: "a group of patterns that provide mechanisms for live operating contracts".
The problem that this pattern aims to solve is simple: smart contracts are immutable. Therefore, changing contract functionality requires the deployment of a new contract The solution is to "outsource functional units that are likely to change into separate satellite contracts and use a reference to these contracts in order to utilize needed functionality.

The satellite pattern allows to modify and replace contract functionality. This is achieved through the creation of separatesatellite contracts that encapsulate certain contract functionality. The addresses of these satellite contracts are stored in a base contract. This contract can then can call out to the satellitecontracts when it needs to reference certain functionalities, byusing the stored address pointers" (Wöhrer and Zdun 2018).
Appendix G exemplifies the application of this pattern.

DFiles Main Smart Contract

Developing Ethereum smart contracts requires a substantial amount of design and testing to ensure they are bug-free; as they cannot be changed once deployed. In DFiles, the core smart contract, **Files.sol**, was developed using the popular Ethereum programming language Solidity and Web3JS for interacting with it in the frontend. From the multiple readily available Solidity design patterns, only the Ownable one was chosen, due to the simplicity of the developed smart contract. Its implementation was provided by OpenZeppelin[8]. The code was inherited in the Files.sol smart contract. Then, the overall data structures to use was decided. For the user files, a mapping to map the user's Ethereum address to a File structure, **userFiles**. This structure stores data about a file, in addition to the user's uploaded file IPFS hash, as seen in Listing 6.1.

```
1  // A structure that contains information about the user's files
2  struct File {
3      string name;
4      string extension;
5      uint32 size;
6      string hash;
7      string timestamp; // the transaction timestamp is stored in the
       JavaScript and is in the Unix Epoch time;
8      address ethereumAccount;
9    }
10
11 //The files belonging to each user (or address)
12 mapping(address => File[]) private userFiles;
```

Listing 6.1: Files.sol smart contract data structures.

[8] https://github.com/OpenZeppelin/openzeppelin-solidity

A function was then developed to add files to the userFiles mapping, as displayed in Listing 6.2.

```
1  /// @notice This function inserts information of a file to the userFiles
       array: name, size, extension in addition to a timestamp and the IPFS
       file hash
2  /// @param _name file name
3  /// @param _hash IPFS file hash
4  /// @param _extension file extension
5  /// @param _size file size
6  /// @param _timestamp a timestamp
7  /// @param _ethereumAccount the user Ethereum account calling this
       contract
8  function addFile(string _name, string _hash, string _extension, uint32
       _size, string _timestamp, address _ethereumAccount)
9  public {
10
11     require(_size>0 && _size<=4294967295, "Invalid file size");
12     File memory file = File(_name, _extension,  _size, _hash,_timestamp,
       _ethereumAccount);
13     userFiles[contractOwner].push(file);
14
15     // emit AddFile event
16     emit AddFile(_name, _hash, _extension, _size, _timestamp,
       _ethereumAccount);
17
18   }
```

Listing 6.2: Adding file information in the Files.sol smart contract.

Finally, two read-only functions were created to return the index of a given file and to return the mapping's length. These allow one to save a substantial amount of gas by iterating the mapping via Web3JS in the frontend. The complete code of the Files.sol source code is available at Etherscan[9] and Appendix I.

Contracts Diagram

In addition to presenting the Files.sol source code, an adaptation of the classes UML diagram, is also recommended for one to fully visualize the smart contracts present in DFiles. Figure 26 is this **contracts** diagram.

[9] https://rinkeby.etherscan.io/address/0x37b143616da99bb2a9bc6f2b01ed4f0872343654#code

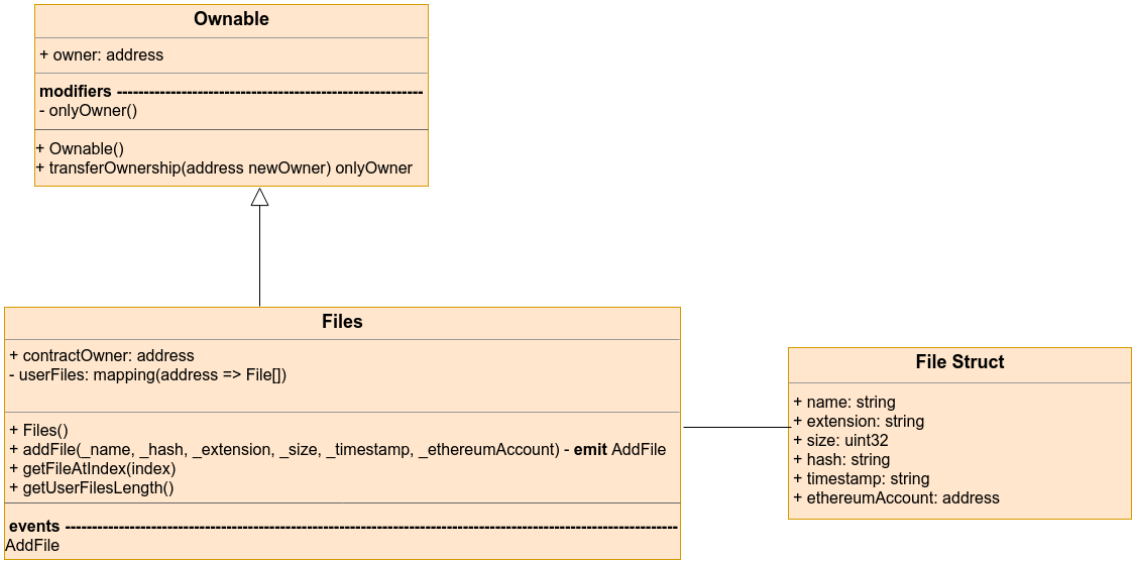

Figure 26: The different smart contracts in the DFiles DApp, in addition to their variables, functions, modifiers and events.

Security

From 2015 onwards, there have been multiple attacks on Ethereum with the major ones being the DAO and Parity hacks. For this reason, a strong focus on security is paramount. In order to build bug-free and secure contracts, the overall process is twofold:

1. Figure out what security Solidity design patterns to use
2. Run static analysis tools. These analyze the code before it is executed to determine if it contains well-known vulnerabilities or other problems in the code.
   In DFiles, the Ownable.sol smart contract is built with audited code by OpenZeppelin and thus, it is secure and bug-free. On the other hand, the other smart contract, Files.sol must be subject to this static analysis in addition to thorough testing (this is discussed in section 6.2).

Note that the Files.sol smart contract is a simple one: it does not receive or send Ether, nor it is used for tokens. For this reason, from all the security design patterns, only the Ownable one was used.

Two tools were used to statically analyze the code: **Securify**[10] and **SmartCheck**[11]. Both are still experimental tools, with the former giving more false positives than the latter. Regardless of this, both are used, as one might detect a serious problem in the code while the other may not.

[10] https://securify.chainsecurity.com/
[11] https://tool.smartdec.net/

Securify

Securify only detected a possible vulnerability in the final version of the Files.sol smart contract: **Unrestricted write to storage**. Figure 27 better describes this possible vulnerability.

Figure 27: The unrestricted write to storage possible vulnerability. From Securify.

After a close review of this possible vulnerability, it can be concluded that Securify is indeed right, it **is** a vulnerability. An attacker can pass on a new address as its owner. This was known when the smart contract was developed, and can be considered more a design choice than a vulnerability.

SmartCheck

SmartCheck is another static analysis tool for Ethereum smart contracts. It was also used to further test the Files.sol smart contract in addition to attempt to reduce significantly the false positive results.

This tool appears to detect more vulnerabilities in the code than Securify. These include:

1. **No payable fallback function**: the smart contract does not have a **payable** fallback and thus, all attempts to transfer and send Ether to this smart contract will be reverted, as shown in Figure 28. This is understandable, as the smart contract only stores the user's uploaded IPFS hash and file metadata and does not need to send or receive Ether

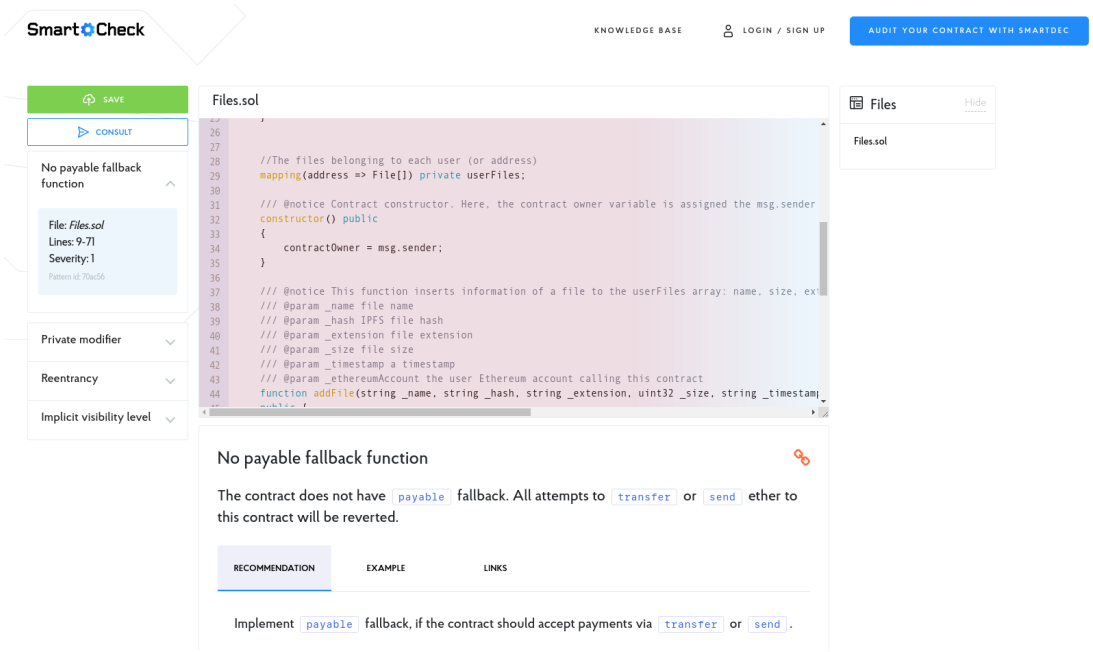

Figure 28: No payable fallback function warning. From SmartCheck.

2. **Private modifier**: the smart contract uses a **private** modifier, as seen in Listing 6.3.

```
1 mapping(address => File[]) private userFiles;
```

Listing 6.3: The usage of the private modifier.

Contrary to other programming languages, the keyword **private** in this case **does not** make a variable invisible. Miners still have access to all contract's code and data. This is one of the reasons files are encrypted with IPFS, as there is a clear lack of privacy in Ethereum. Figure 29 illustrates this.

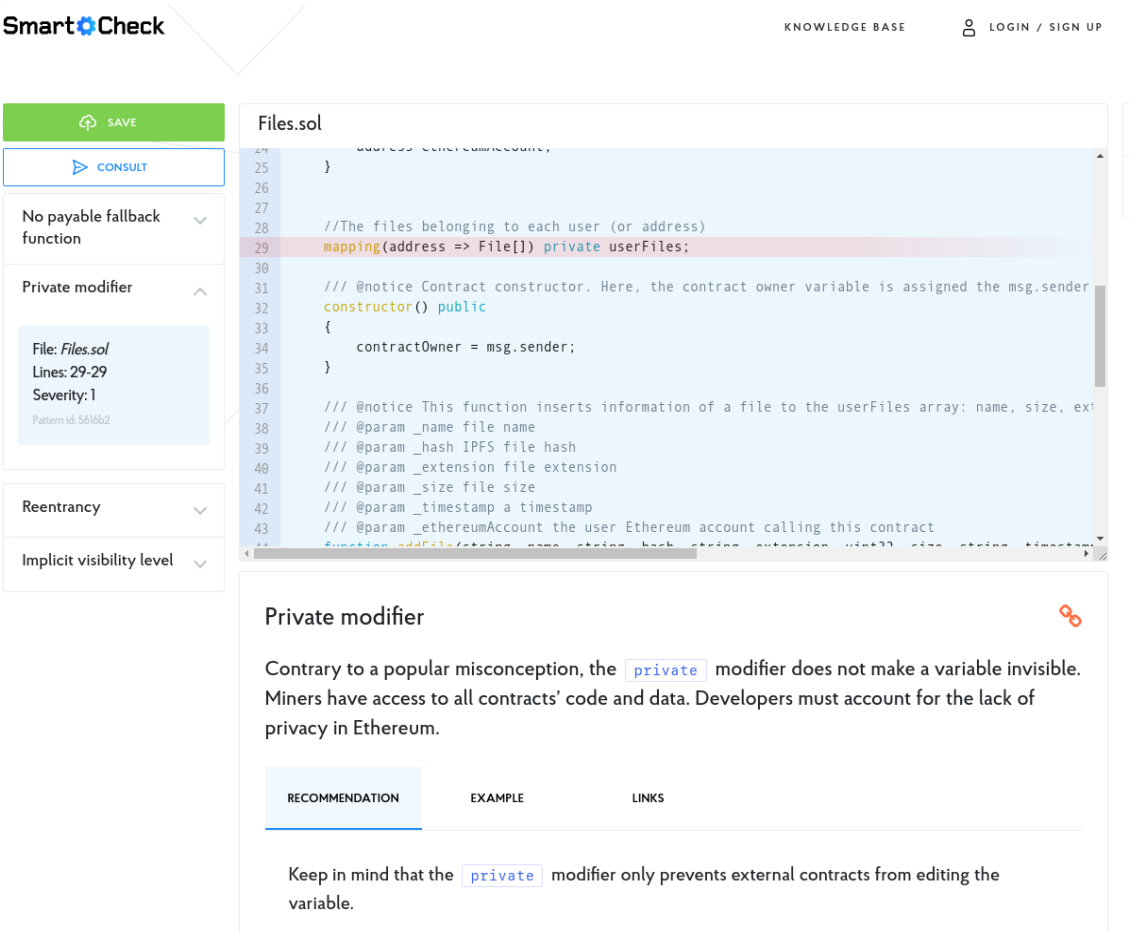

Figure 29: SmartCheck private modifier warning. From SmartCheck.

3. **Reentrancy**: this is a common problem in Ethereum smart contracts, when functions in a smart contract A can be called repeatedly, before the first invocation of the function is finished. This may cause different invocations of the same functions to in interact in unpredictable and destructive ways. SmartCheck reports this possible problem for a specific line in the smart contract code, as presented in Listing 6.4.

```
1 userFiles[contractOwner].push(file);
```

Listing 6.4: The line of code possibly affected by reentrancy.

Figure 30 shows the complete explanation of this possible vulnerability. However, the smart contract does not, as previously stated, send nor receive Ether. For this reason, it can be assumed SmartCheck provided a false positive.

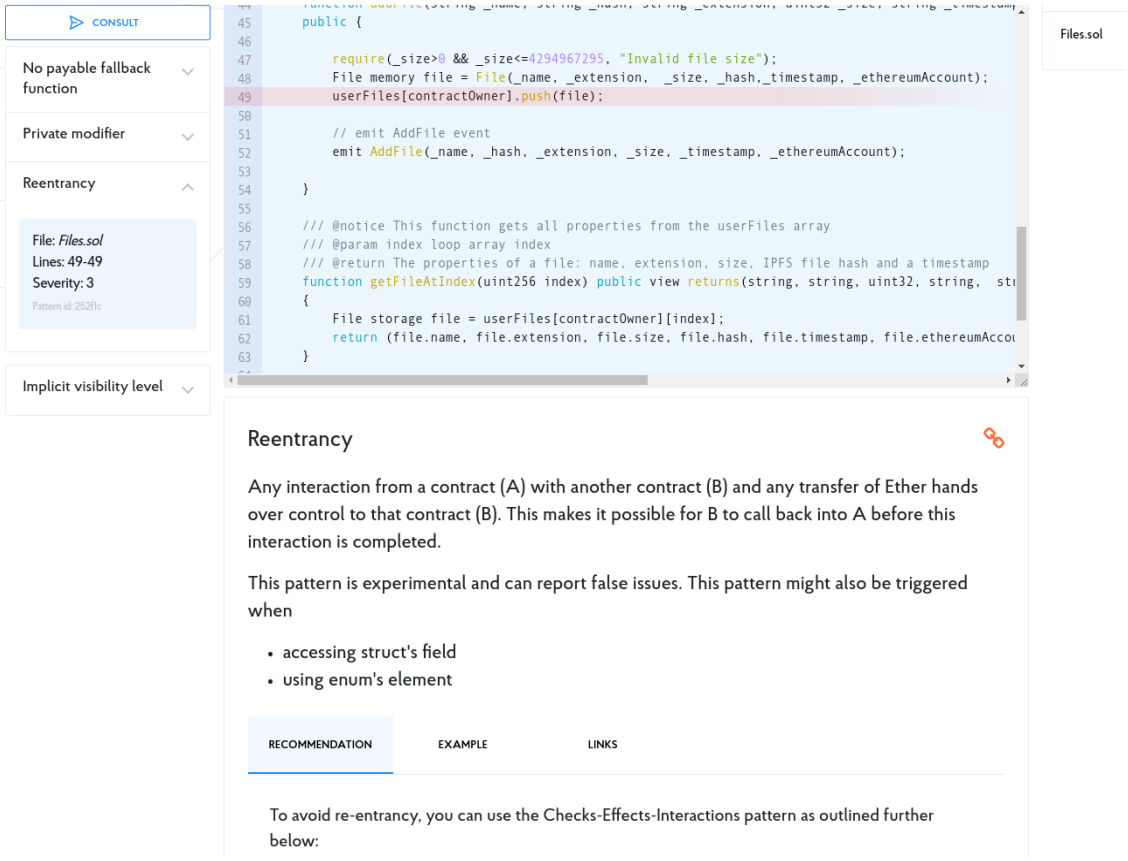

Figure 30: SmartCheck possible reentracy vulnerability. From SmartCheck.

4. **Implicit visibility level**: in a specific line of code (Listing 6.5), SmartCheck warns about a minor aspect, the fact that the default visibility level in Solidity is public. Thus, function visibility should always be defined (private, public, etc).

```
1 address contractOwner public;
```

Listing 6.5: Fixing the implicit vulnerability level in the Files.sol smart contract.

Figure 31 displays this suggestion.

Figure 31: SmartCheck implicit vulnerability level. From SmartCheck.

In conclusion, security in smart contracts is of extreme importance when developing any robust DApp. For this reason, identifying the multiple design patterns to use (in addition to the ones focused on security) is a great step towards developing bug-free, safe and secure smart contracts. On top of this, there are two important static analysis tools: **Smartcheck** and **Securify**. Both provide valuable input when developing smart contracts, as vulnerabilities can be easily detected and fixed. This is critically important, as any smart contract is immutable.
As for Files.sol, in DFiles, this smart contract appeared to contain multiple vulnerabilities. However, some were false positives while others were design choices, even if they can be exploited. With all this taken into consideration, only a minor tweak could be made in the smart contract's code, by always specifying the function vulnerability level, as displayed in Listing 6.5.

6.1.2 Frontend

In the previous section, the design and implementation of the DFiles DApp was documented while adhering to the Blockchain Software Engineering. In this section, the frontend aspect of the DApp is detailed. Note that the frontend is not as important as the backend, and therefore will only be briefly elaborated. The frontend uses three main technologies:

- **Bootstrap** and **Flexbox** for easy responsive design
- **HTML5** and **SASS** for displaying content on screen and styling it
- **ReactJS** as the main engine for developing the User Interface. It combines the four previous technologies to display data on screen

As previously stated, ReactJS is the core tool to build elegant, simple and easy-to-use User Interfaces. In this section, the following is addressed:

- **React Router**[12]: as a collection of ReactJS components, it allows for developers to specify web application routes, with NodeJS and Express. Following the same train of thought, the defined routes for the DFiles DApp are presented here
- **Folder Structure**: ReactJS does not offer a standardized method for organizing folders[13]. For this reason, a custom folder structure was designed and is documented here

Folder Structure

The official ReactJS docs[14] present two viable options for the folder structure:

- Grouping by features or routes, where files are located inside folders grouped by feature or route
- Grouping by file type, where files with the same extension are grouped in the same folder

However, the approach taken for DFiles was not any of the above methods. Instead, a combination of grouping by features or routes and special folders for reusable components, styles, authentication and folders, as such:

- **auth**: the login and register webpages are listed here
- **components**: every reusable component, ranging from buttons, to HTML tables, to the Navbar, are listed in this folder in addition to SASS files for their styling
- **fonts**: all of the fonts required for the DApp are listed here
- **img**: the images used for the DApp are shown here
- **pages**: a decision was made to group webpages by route:
  - **home**: represents the homepage, with all its components joined together (footer, header and main)
  - **files**: everything related to uploading and viewing a user's files is listed here:
    - **my files**: webpage to display the user's uploaded files, plus its styling
    - **upload files**: the same as above, but for uploading the user's files to the Ethereum Blockchain
  - **my account**: webpages responsible for displaying information about a user's account, such as his email or Ethereum Address, in addition to his GDPR rights (right to erasure, right to rectification, etc)
- **styles**: the different styles for the multiple webpages and components are listed in this folder. Moreover, SASS style rules for multiple devices are coded here, to ensure the DApp displays properly in all devices — responsive design

Figure 32, below, further illustrates this folder structure.

[12] https://reacttraining.com/react-router/
[13] https://reactjs.org/docs/faq-structure.html
[14] https://reactjs.org/docs/faq-structure.html

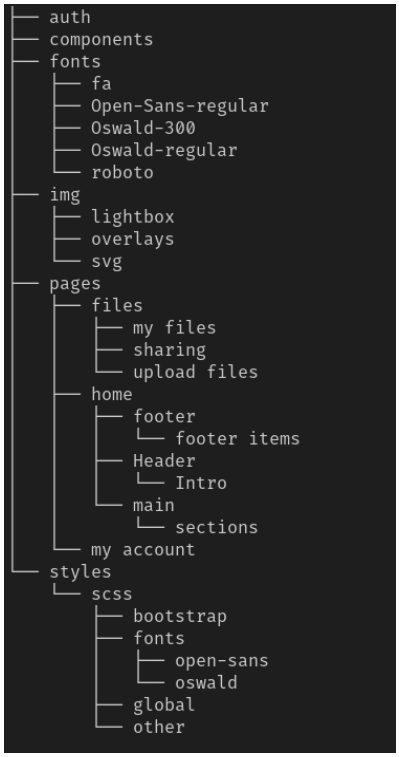

Figure 32: ReactJS folder structure.

Note that SASS and Bootstrap are not explicitly documented. The reasoning behind this is that this work does not focus on frontend development and as such, only the basic details are presented.

React Router

React Router allows developers to specify in a direct, easy way routes to facilitate the flow of information in any web application. Therefore, defining the key routes a web application must take, is crucial to its development.
The DFiles DApp has the following routes, as defined in Table 32.

Table 32: React Router routes.

Route	Component	Description
/auth/account	/pages/my account/MyAccount	Displays information about the user's account
/auth/account/settings	/pages/my account/MyAccountSettings	Displays the user's account settings
/auth/account/ settings/privacy	/pages/my account/MyAccountPrivacyAndGDPR	Presents and displays specific information regarding the GDPR to the user
/auth/login	/auth/Login	User login page
/auth/register	/auth/Register	User registration page
/files/upload-files	/pages/files/upload files/UploadFiles	File upload page which allows users to encrypt and upload files to the IPFS with its metadata and generated hash stored in the Ethereum Blockchain
/files/my-files	/pages/files/my files/MyFiles	Allows users to decrypt and view their uploaded files from its IPFS hash in addition retrieving its metadata from the Ethereum Blockchain

6.2 Testing

When developing DApps, testing can be considered the most important phase of the software development lifecycle, as smart contracts are immutable, and thus cannot be changed. For this reason, they must be thoroughly tested, where every scenario is thought out carefully for decades to come. Unlike standard software engineering, where fixes can be applied later on to software, in Blockchain Software Engineering, testing in smart contracts can only be done until deployment. After this happens, these are immutable for decades to come and patches are almost impossible to be issued.

For the reasons mentioned above, when developing smart contracts for the DFiles DApp, testing was taken seriously, with built-in tools in the Truffle Framework (Mocha and Chai) and Ganache, for local testing. In the overall project, there is a special folder for Truffle unit testing with Web3JS: the **test** folder. Inside, only one file was created to test the Files.sol Ethereum smart contract.

There were two main tests performed to attempt to thoroughly test the Files.sol Ethereum smart contract:

1. Creating a JSON object which simulates a user's uploaded file:

- **name**: file name
- **extension**: file extension
- **size**: file size, in bytes
- **hash**: the fake IPFS hash, as if the user had uploaded a file to an IPFS node and it had returned a hash to the file
- **timestamp**: file uploaded timestamp
- **ethereumAddress**: the Ethereum address the simulated file belongs to

After this is created, it was attempted to then insert it in the local Ganache Ethereum Blockchain, by invoking the **addFile** function in the smart contract and passing the above created JSON object. The final step is to loop through this file, by using other functions in the contract such as **getFileAtIndex** in our local, Ganache Blockchain. It is then asserted if the file inserted in the local Blockchain, after looping through it, is a match to the test JSON object mentioned above. If this is true (meaning the object above was inserted and looped through correctly in the local Blockchain), then the test passes. Otherwise, something seriously went wrong when inserting the file into the local Blockchain. Listing 6.6 has the code for this test.

```
1  it("should be possible to store a file's properties in the Blockchain
       !"
2      // And it should have a length of 1"
3      , async () => {
4          await filesContractInstance.addFile(
5              filesNotFromBlockchain[0].name,
6              filesNotFromBlockchain[0].hash,
7              filesNotFromBlockchain[0].extension,
8              filesNotFromBlockchain[0].size,
9              filesNotFromBlockchain[0].timestamp,
10             filesNotFromBlockchain[0].ethereumAddress, {
11                 from: accounts[0]
12             }
13         );
14
15         let fileRetrievedFromSmartContract = await
       filesContractInstance.getFileAtIndex(
16             0
17         );
18
19
20         const fileFromBlockchain = {
21             name: fileRetrievedFromSmartContract[0],
22             extension: fileRetrievedFromSmartContract[1],
23             size: fileRetrievedFromSmartContract[2].toNumber(),
24             hash: fileRetrievedFromSmartContract[3],
25             timestamp: fileRetrievedFromSmartContract[4],
26             ethereumAddress: fileRetrievedFromSmartContract[5],
27         };
28
29         assert.equal(
30             JSON.stringify(filesNotFromBlockchain[0]),
31             JSON.stringify(fileFromBlockchain),
32             "File was not added correctly to the Blockchain :("
33         );
34
```

```
35          const filesNotFromBlockchainLength = filesNotFromBlockchain.
    length.toString();
36          const fileFromBlockchainLength = await filesContractInstance.
    getUserFilesLength();
37          assert.equal(
38              filesNotFromBlockchainLength,
39              fileFromBlockchainLength.toString(),
40              "Something is wrong, there should be ONLY one added file
    in the Blockchain :("
41          );
42      });
```

Listing 6.6: Unit test to assert if data about a file and its IPFS hash can be inserted successfully into the Ethereum Blockchain (locally).

2. The second test is to attempt to loop through an invalid index in the **userFiles** mapping in the Files.sol smart contract. It is expected that this looping will fail, as there will not be such a file. The ultimate goal is to assert if this happens, or if actually something went seriously wrong while either looping though the simulated uploaded files to the local Blockchain or if everything went according to plan and there is no such file, even after browsing through all the files in the Blockchain. Listing 6.7 contains the code for this unit test.

```
1   it("should NOT be possible to retrieve nonexistent files from the
    Blockchain!!!", async () => {
2     const invalidFileIndex = 99;
3
4     await filesContractInstance.addFile(
5       filesNotFromBlockchain[0].name,
6       filesNotFromBlockchain[0].hash,
7       filesNotFromBlockchain[0].extension,
8       filesNotFromBlockchain[0].size,
9       filesNotFromBlockchain[0].timestamp,
10      filesNotFromBlockchain[0].ethereumAddress, {
11        from: accounts[0]
12      }
13    );
14
15    try {
16      await filesContractInstance.getFileAtIndex(invalidFileIndex);
17      assert.ok(true, `file  is valid. `);
18    } catch (error) {
19      assert.equal(
20        error,
21        `Error: VM Exception while processing transaction: invalid
    opcode`
22      );
23    }
24  });
```

Listing 6.7: Unit test to assert if looping through an invalid file index should fail or not.

These two tests produce the expected results, when using Truffle, as shown in Figure 33.

```
[miniclip22@ACDCROCKS-DESKTOP-ARCHLINUX:tmdei-2017-2018-ethdocstore-ethereum|master **]$ $truffle test
Using network 'development'.

accounts:  [ '0x84f3aa80789b9995ee56804cf86d057ec769b96c',
  '0x22bc5e1d9ed059184d43912e6c322496a156e748',
  '0x4858d3e1ae140f3897657922d72c3c35e800ed60',
  '0xc7fda6c07c6ff2a093d11bf71c95cdca7b139680',
  '0x5fc1c605cc36431722674ec15eba443cdd4d488e',
  '0xd36d0e01e468c982da177a2e9f714935702213aa',
  '0x8d865e755f21eb1963b8104e497c6b164ec5cfd0',
  '0x5112cc82d6244f82fcda27c968f49dbcf904647e',
  '0x03e61b9885d8e7a537afb8bc7c21efcf22fcfde0',
  '0xa15297742d0f7451092dfed96613377f8d6a1e88' ]


  Contract: Files
    √ should be possible to store a file's properties in the Blockchain! (107ms)
    √ should NOT be possible to retrieve nonexistent files from the Blockchain!!! (130ms)


  2 passing (265ms)
```

Figure 33: The overall result of running unit tests in Truffle.

The complete file for the above unit tests is available at Appendix J.

6.3 Deployment

Deployment is the last phase in Blockchain Software Engineering and can also be considered one of the hardest. In traditional software applications, typically they are migrated to one or more servers, and within a few minutes the whole system is up and running. In Ethereum DApps, this process is completely different. To begin with, each node in the Ethereum Blockchain must have the same copy of all transactions until a certain point. Moreover, there is the special issue of immutability in Ethereum smart contracts; these must be thoroughly tested with every scenario carefully thought for decades to come. For this reason, their deployment is a risky process: there might be an undetected bug in the code, which ultimately may allow an attacker to disrupt the whole system or worse, steal all the cryptocurrency to another Ethereum wallet, if that is the case.
On top of all these reasons, deployment in DApps is also a costly process: every computational step must be paid when deploying smart contracts. They have to be as efficient as possible, without sacrificing security.
In this section, only the deployment of Ethereum smart contracts (the centralized component of DFiles is not detailed here) to the Rinkeby Ethereum test network is explained, with the two best approaches presented, in addition to their advantages and disadvantages. Both use the tools present in the Truffle Framework to facilitate deployment.

6.3.1 Local Ethereum Rinkeby Node

The files.sol Ethereum smart contract can be deployed to the Rinkeby test network by either running a local Rinkeby Ethereum node, or via an Infura Rinkeby node. This first approach to deployment relies on the former.
By analyzing the official Rinkeby documentation[15], a local Rinkeby node can be run by using

[15] https://www.rinkeby.io/#geth

geth[16] and first downloading the **rinkeby.json** file[17]. Then, a set of commands must be run, as displayed in Listing 6.8 and Listing 6.9.

```
geth --datadir=$HOME/.rinkeby init rinkeby.json
```

Listing 6.8: The first command to setting up a local Rinkeby node.

```
geth --networkid=4 --datadir=$HOME/.rinkeby --cache=512 --ethstats='
  MyAwesomePC:Respect my authoritah!@stats.rinkeby.io' --bootnodes=enode:
  //a24ac7c5484ef4ed0c5eb2d36620ba4e4aa13b8c84684e1b4aab0cebea2ae45cb4d37
  5b77eab56516d34bfbd3c1a833fc51296ff084b770b94fb9028c4d25ccf@52.169.42.1
  01:30303
```

Listing 6.9: The second command to setting up a local Rinkeby node.

After running the above commands, the local node starts syncing with the Rinkeby tesnet, as displayed in Figure 34.

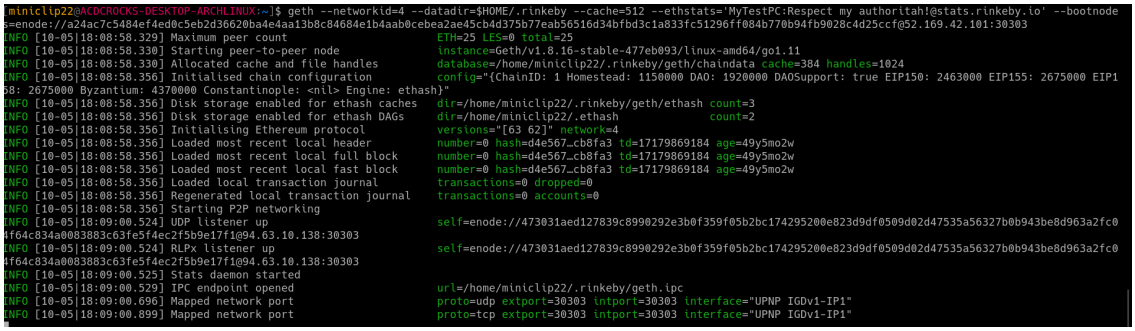

Figure 34: The local Rinkeby Ethereum node syncing.

After a few hours and around 15GB, the local node is then fully synced with the latest block in the Rinkeby Ethereum test network. However, the smart contracts deployment cannot begin just yet: the Truffle framework must be configured first, in the file **truffle-config.js**. This file can contain multiple configurations for other tesnets or the Ethereum main network. Listing 6.12 displays the multiple network configurations for this first approach.

```
module.exports = {
  networks: {
    development: {
      /* Ganache */
      host: "localhost", // Host name
      port: 7545, // Host port
      network_id: "*" // Match any network id
      },
      rinkebyLocalNode: {
        host: "localhost", // Host name
        port: 8545, // Host port
```

[16] https://github.com/ethereum/go-ethereum/wiki/geth
[17] https://www.rinkeby.io/rinkeby.json

```
12            from: "0xaa6c153202ac204dae3c62a3e5bc0b851f10886d", // The
    Ethereum Rinkeby address for the Truffle Framework to make transactions
13            network_id: 4, // Rinkeby Testnet, 1 is for the main one
14                gas: 4612388 // Gas limit used for deploys
15      }
16    }
17 };
```

Listing 6.10: The multiple network configurations in **truffle-config.js.**

Once both the local Rinkeby node and the Truffle Framework are configured for deployment, the next stage is in fact, to deploy the developed smart contracts (in DFiles, these are the Ownable.sol and Files.sol). To do this, one must request to the Rinkeby faucet some Ether to pay for all the computational steps required in the deployment, as shown in Figure 35. Note that this Ether does not have any value, it is simply used as a measure to prevent abuse in the network.

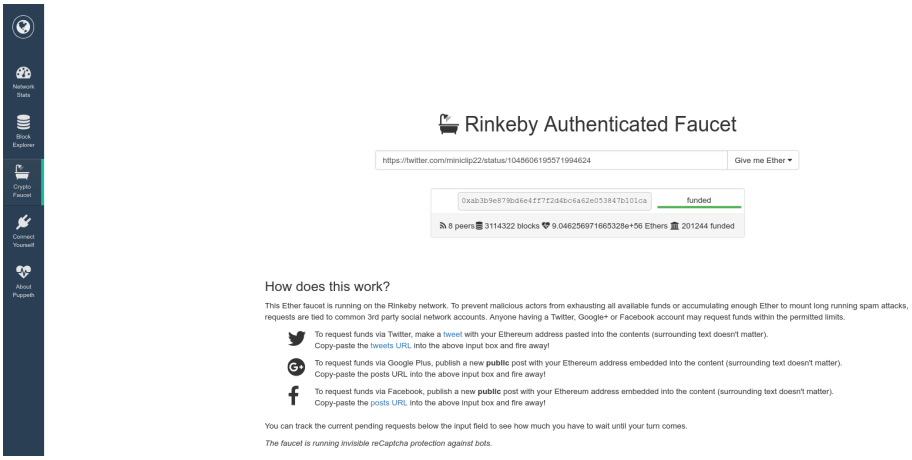

Figure 35: ARinkeby Faucet funding Ether into Ethereum address 0xab3b9e879bd6e4ff7f2d4bc6a62e053847b101ca.

Once Ether has been received from the Faucet, the development process can therefore begin. First of all, the **truffle compile** command must be executed to ensure all the Ethereum smart contracts are in fact error-free and available for deployment. After this command, a special Truffle migrations file must be created to include the desired smart contracts for deployment (folder **migrations**): Listing 6.11.

```
1
2 let Files = artifacts.require("./Files.sol"); // Import the contract to be
      deployed
3
4 module.exports = function (deployer) {
5   deployer.deploy(Files); // tell Truffle to deploy the Files.sol smart
      contract
6 }
```

Listing 6.11: The special **migrations** file used to specify which smart contracts are to be deployed.

Figure 36 contains the Truffle command and its output, after a successful deployment (the Ganache personal Ethereum Blockchain was used in the figure, not the local Rinkeby node).

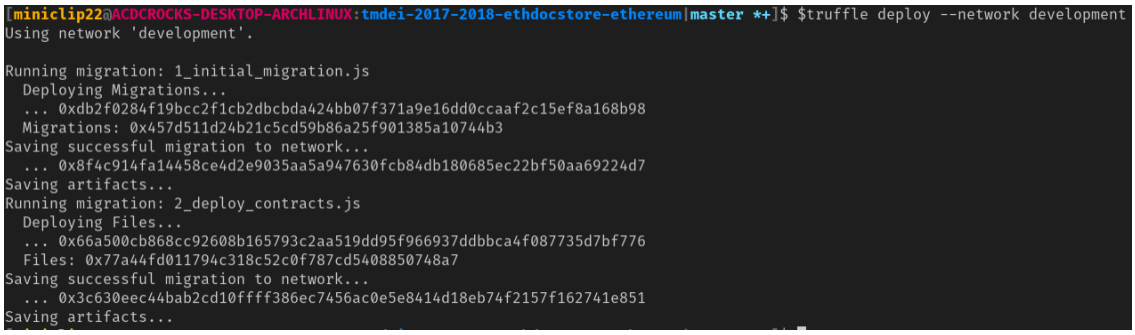

Figure 36: An example of a successful deployment.

The end result is a publicly available list of transactions as a result of this deployment. As this approach was not used for the deployment of the Ethereum smart contracts, this will be explained in the end of that method of deployment.

6.3.2 Infura Rinkeby Node

The second approach for the deployment considered is the usage of the Infura Rinkeby node. To begin with, this brings an array of advantages, as the need for a local Rinkeby node disappears, along with the time required to keep it in-sync in addition to the disk space that was needed in the local Rinkeby node. However, the process for deployment is slightly more complex, and trust must be placed in Infura, a centralized organization like metamask.
Instead of syncing a local Rinkeby node, Infura requires that an individual to be registered with them[18] where it generates an unique API key and a mnemonic for one to use its services. They provide in addition to Rinkeby nodes, other Ethereum test networks in addition to the main Ethereum network and IPFS nodes.

The first step is configuring the same **truffle-config.js** file, albeit now for Infura with some slight modifications. To begin with, Truffle's **HDWalletProvider** must be installed and used (npm install truffle-hdwallet-provider)[19]. Then, the Infura's API key generated for a specific account must be declared in addition to the Infura mnemonic. For security reasons, these are passed via system environment variables, and not read from a specific file. Listing 6.12 contains the full configuration for the **truffle-config.js** file, for both Infura and local nodes.

```
1 var HDWalletProvider = require("truffle-hdwallet-provider");
2
3 const mnemonicLocalhost = process.env.mnemonicLocalhost;
4 const mnemonicInfura = process.env.mnemonicInfura;
5 const infuraAPIKey = process.env.infuraAPIKey;
6
7 module.exports = {
8   networks: {
```

[18] https://infura.io/register
[19] https://truffleframework.com/tutorials/using-infura-custom-provider

```
9     development: {
10      /* Ganache, local Blockchain */
11      host: "localhost", // Host name
12      port: 7545, // Host port
13      network_id: "*" // Match any network id
14    },
15    rinkebyInfura: {
16    /* Rinkeby Infura node, used for deployment */
17      provider: function () {
18        return new HDWalletProvider(  // the HDWalletProvider is used with
     the Infura mnemonic and API key passed as system environment variables
19          process.env.mnemonicInfura,
20          'https://rinkeby.infura.io/' + infuraAPIKey
21        )
22      },
23      network_id: 4 // Rinkey contains the network id 4, the Main Ethereum
     one the network id of 1
24      from: "0xaa6C153202Ac204dAE3C62a3E5Bc0b851F10886D", // the Ethereum
    Rinkeby account that Truffle uses for deployment
25    }
26  }
27 };
```

Listing 6.12: The multiple network configurations in **truffle-config.js.**

Truffle is now ready to deploy the developed smart contracts. If required, Ether must be requested to the Rinkeby Faucet (Figure 35). Assuming the chosen Ethereum account for deployment contains enough Ether, then the deployment process may begin. First of all, the **truffle compile** command must be executed to ensure all the Ethereum smart contracts are in fact error-free and available for deployment. After this command, a special Truffle migrations file must be created to include the desired smart contracts for this task: Listing 6.11.

Assuming that all desired smart contracts are compiled successfully and specified for deployment, Figure 37 contains the command and output of a successful deployment to an Infura Rinkeby node.

```
[miniclip22@ACDCROCKS-DESKTOP-ARCHLINUX:tmdei-2017-2018-ethdocstore-ethereum|master **]$ $truffle deploy --network rinkebyInfura
Using network 'rinkebyInfura'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x457b85634ebe25794733071782121abf24a92b5ea2bab9138d04a0c80b725c5d
  Migrations: 0xce3a8cf8083c848a806450a24ea0b8f51762b5aa
Saving successful migration to network...
  ... 0xbbf2e0ceed8603ad69039bb8657f7f5f3b34f3e469054dcca4120ab2a1339ca0
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Files...
  ... 0xd40a490172c52e8e0c6fcbe4c9cb041a8fda12082c580f74c9df7d7ed91de47b
  Files: 0xfd6168d9f156e9e07ee6bc31c6ad357a32ec1f76
Saving successful migration to network...
  ... 0x5bebf8678c660318b1c651511eacdb5474c1f1f1abb41e25aef995a930020560
Saving artifacts...
```

Figure 37: An example of a successful deployment with an Infura Rinkeby node.

After this process, the transactions can be viewed on Etherscan[20]: Figure 38.

[20] https://rinkeby.etherscan.io/address/0xfd6168d9f156e9e07ee6bc31c6ad357a32ec1f76#code

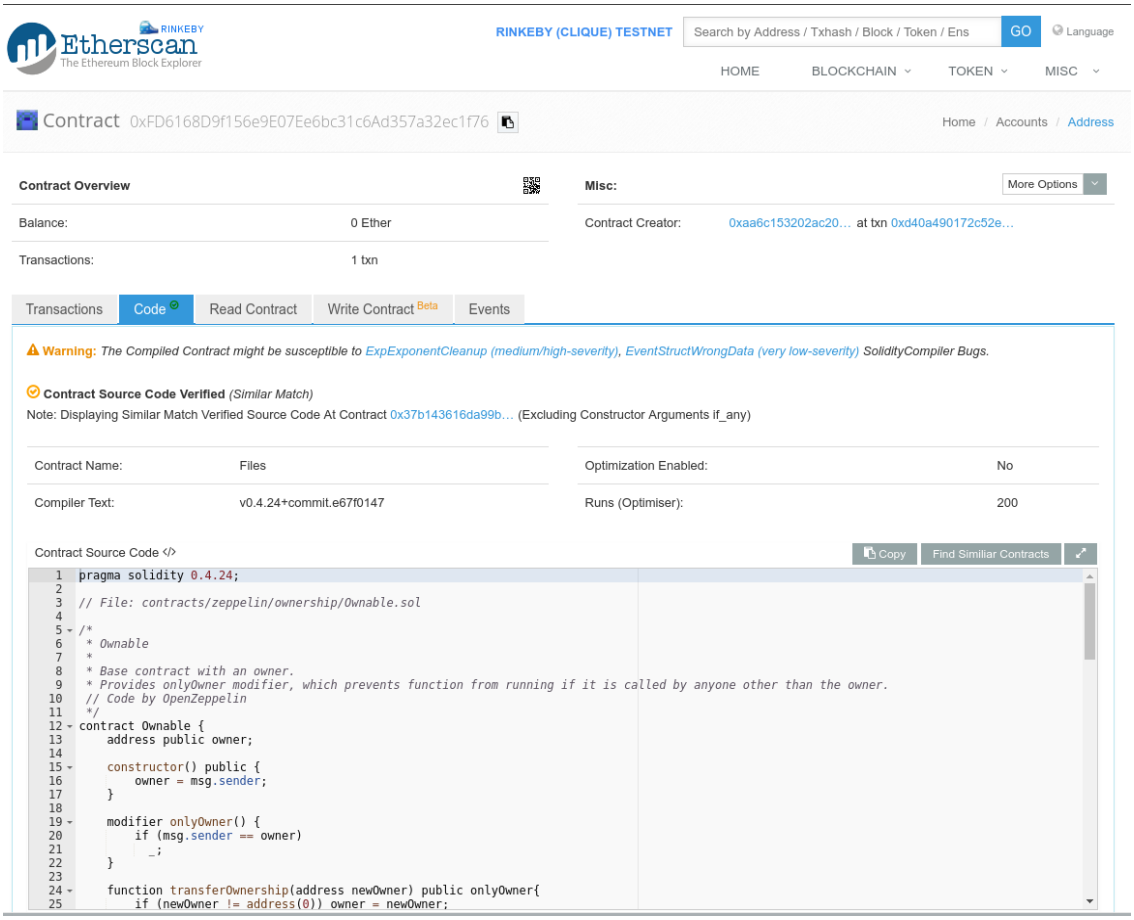

Figure 38: The Etherscan Rinkeby page for the Files.sol deployed smart contract.

Etherscan also allows anyone to interact with deployed smart contracts, by using the **Read Contract** and **Write Contract** tabs, where their content can be read and written to, respectively. Figure 39 shows a possible output for **Read Contract** tab (it also uses metamask configured for the Rinkeby test network).

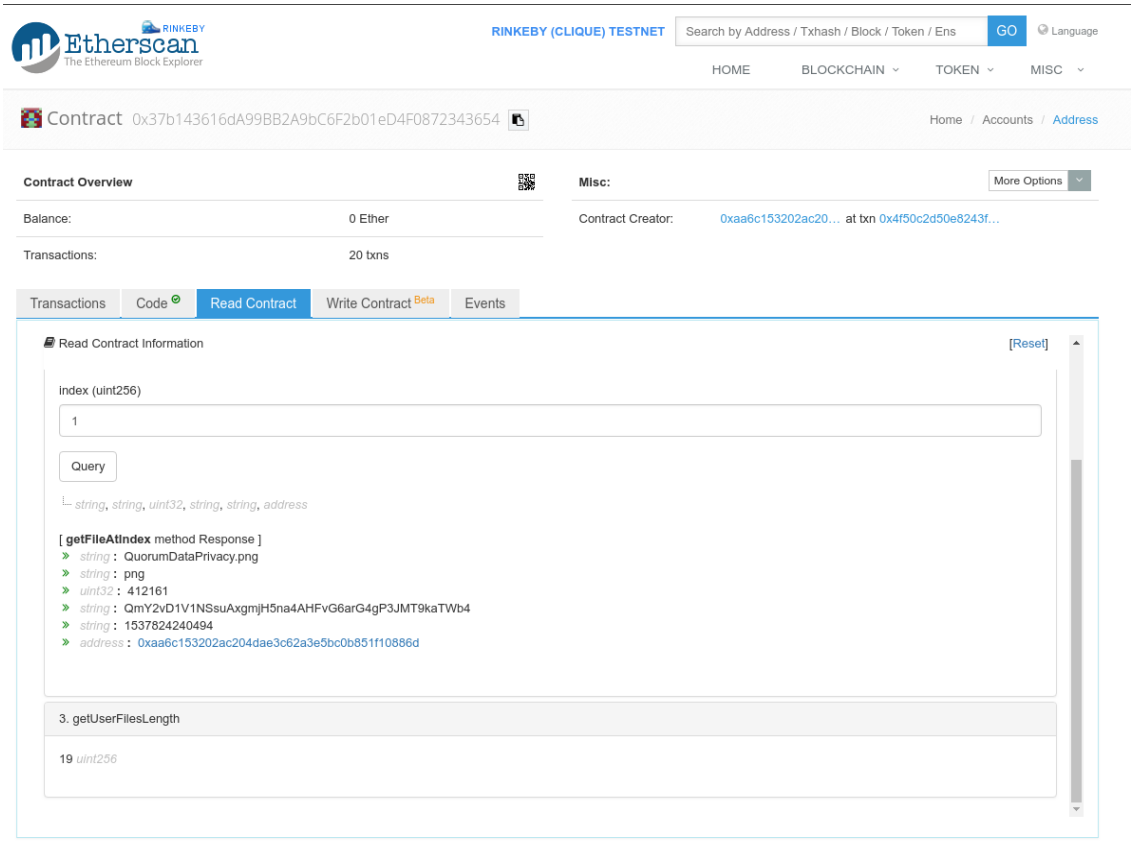

Figure 39: The **ReadContract** tab output in Etherscan, for the Rinkeby test network.

Etherscan is a great tool for anyone to view all the transactions made to and from a given smart contract. However, this also represents a major privacy problem. This is the main reason why private data is encrypted by using an IPFS node.

On a final note, both approaches are valid for deploying Ethereum smart contracts. Both have advantages and disadvantages. For instance, the local Rinkeby node needs to sync with the network, which in turn needs at least 15GB of free storage space for this task and requires a few works to be fully synced. This can be seen as a major disadvantage of this method. However, this node is one that can be easily controlled and does not rely on a third party such as Infura. Configuration only requires a few commands to be begin the syncing process for the local node. Thus, this is also a major advantage towards setting up deployment with an Infura node.

The Infura node brings some key advantages. To begin with, disk space and syncing problems are solved, as the company is responsible for its syncing and maintenance. However, this implies trust must be placed in an external third party, Infura. They can, without further notice, attempt to disrupt the deployment process, albeit highly unlikely. Also, as previously mentioned, configuration is slightly more complicated and requires an account with Infura. After a careful analysis of the pros and cons mentioned above, the Infura node was chosen for the deployment due to the fact it does not require precious disk space and syncing with

the Rinkeby Ethereum tesnet and that deployment times with each method are theoretically identical (the first method of deployment was not used, but it can be assumed they are in fact identical).

Chapter 7

Data Protection

Data protection and its subset data privacy is of extreme importance in the modern world. Without them, large corporations enrol in a free-for-all mentality towards the important and profitable data provided by their customers. In recent years, companies have employed bad practices regarding their customer's personal data, with some dire consequences in specific cases.

The European Union has been trying to counter this tendency for over two decades. In 1995, it introduced the Data Protection Directive[1] aimed at regulating the processing of personal data within the European Union.
However, this regulation proved to be insufficient in efficiently protecting personal data in the last two decades.

To better protect EU citizens and their personal data, a new regulation was created in April 2016 — the General Data Protection Regulation — and enforced May that year (the 1995 Data Protection Directive was repealed). However, there were no penalties being distributed until two years later, on May 25th, 2018.
The GDPR has given individuals control over what kind of personal data is processed, collected and stored with new rights for individuals and obligations for organizations with offices in EU territories, even if their headquarters are resided outside the EU.

In this chapter, first a subset of the official documentation of the GDPR is presented, with the main rights and obligations required for the context of this work. Secondly, a plan for most of DApps to achieve GDPR compliance is provided. After this, a specific plan is drafted for the DFiles DApp to comply with this regulation in addition to its possible implementation. Finally, a section about the regulation's key changes.

7.1 General Data Protection Regulation

Today, the GDPR might be considered the most important regulation to protect individuals' personal data and ensure organizations are complying with certain obligations. In this section, only a subset of the most relevant and important parts of the GDPR are detailed here. However, the data protection by design and default principles are not covered in this

[1] https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:31995L0046

work.

Also note that this does not replace the official GDPR guidelines[2].

Before detailing the ins and outs of this complex regulation, some key definitions are presented, albeit not always applicable. For the nature of this work, the following definitions are used (Article 4):

- **Data subject:** an individual whose personal information is being collected, processed or held (EU GDPR Compliant 2018)
- **Personal data:** "any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person" (European Union 2016)
- **Processing:** "any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organization, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction" (European Union 2016)
- **Profiling:** "the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person" (European Union 2016)
- **Controller:** "the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data; where the purposes and means of such processing are determined by Union or Member State law, the controller or the specific criteria for its nomination may be provided for by Union or Member State law" (European Union 2016)
- **Processor:** "a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller" (European Union 2016)
- **Consent of the Data Subject:** "any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her" (European Union 2016)
- **Personal Data Breach:** "a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed" (European Union 2016)

[2] https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN

7.1.1 Data Controllers and Data Processors

The GDPR applies to both controllers and processors. A controller determines the purposes and means for processing personal data, whereas a processor is responsible for processing personal data on behalf of the controller (ICO 2018). Data processors have legal obligations such as being required to maintain records of personal data and processing activities; they also have legal liability if a breach occurs.

The GDPR was built with several key principles, such as:

- **Lawfulness, fairness and transparency**: according to the regulation's Article 5: "personal data shall be processed lawfully, fairly and in a transparent manner in relation to the data subject" (European Union 2016). In other words, there are four key obligations to this principle to keep in mind (ICO 2018):
  1. Valid grounds for collecting and using personal data must be identified
  2. Personal data must not be used to breach other existing laws
  3. Personal data must not be processed in a way that is unduly detrimental, unexpected or misleading
  4. Openess and honesty are required when using a data subject's personal data
- **Purpose limitation**: in Article 5, purpose limitation is described as: "personal data collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes" (European Union 2016). Furthermore, this translates to the following key ideas (ICO 2018):
  1. The purposes for data processing must be clear from the beginning
  2. The purposes of data collection and specify these to individuals in the privacy policy
  3. Personal data can be used for a new purpose if either this is compatible with the original purpose and consent is provided, or there is a clear basis in law
- **Accuracy**: according to Article 5, personal data shall be: "accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that personal data that are inaccurate, having regard to the purposes for which they are processed, are erased or rectified without delay" (European Union 2016). To elaborate further on this principle, the following must be considered (ICO 2018):
  1. Steps should be taken to ensure the personal data being held is not incorrect or misleading
  2. Steps must be taken to ensure updated personal data remains correct and therefore is not misleading
  3. Challenges that might pose to the accuracy of personal data must be considered
- **Storage limitation**: this principle can be explained in four straightforward steps (ICO 2018):

1. Personal data must not be kept for longer than needed
2. Justification for how long personal data is kept must be provided
3. Periodic reviews of personal data must be conducted, with its erasure or anonymization when no longer needed
4. Data retention challenges must be carefully considered, as the data subject has the right for data erasure

- **Integrity and confidentiality (security)**: The integrity and confidentially principle can be explained in one sentence (ICO 2018): Appropriate security measures must be in place in order to protect the held personal data.
- **Accountability**: The last principle, accountability, is mainly to ensure one takes responsibility for the usage of personal data and how one can comply with other principles, in addition to taking appropriate measures and records in place to demonstrate such compliant (ICO 2018)

7.1.2 Lawful Basis for Processing

While processing data, there must be a lawful basis for processing personal data. At least one of the six, equally important, lawful basis must apply when processing personal data, as stated in Article 6 (European Union 2016):

- **Consent:** is one form of lawful basis which is often used. Individuals have real choice and control, which ultimately may build trust and engagement, thus enhancing an organization's reputation. However, consent cannot be given by default with pre-ticked checkboxes or other default consent methods. It must be clear, concise and explicit (ICO 2018). Moreover, individuals must be able to withdraw consent at any point in time, as stated by Article 7:

  "The data subject shall have the right to withdraw his or her consent at any time. The withdrawal of consent shall not affect the lawfulness of processing based on consent before its withdrawal. Prior to giving consent, the data subject shall be informed thereof. It shall be as easy to withdraw as to give consent" (European Union 2016).

  Finally, consent should be in consistent review and be refreshed if new events arise (ICO 2018).

- **Contract:** Article 6(1)(b) is another form of lawful basis when processing an individual's data. It is best suited in the following scenarios (ICO 2018):
  - When fulfilling contractual obligations
  - Before defining a contract
  - When processing is strictly necessary
- **Legal obligation:** is another lawful basis if there is a need to process personal data to comply with a common law or statutory obligation (ICO 2018). As in Article 6(1)(c) (European Union 2016):

"Processing is necessary for compliance with a legal obligation to which the controller is subject".

- **Vital interests:** is the lawful basis where processing personal data might protect someone's life. However, in health data, this lawful basis cannot be relied upon if the data subject is capable of giving consent (ICO 2018). Or, as in Article 6(1)(d): "Processing is necessary in order to protect the vital interests of the data subject or of another natural person" (European Union 2016).
- **Public task:** according to Article 6(1)(e), "processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority vested in the controller". This can apply to either when carrying a specific task in the public interest or exercising official authority. In either case, they must be laid down by law (ICO 2018). The individual rights to data portability and erasure, do not apply here.
- **Legitimate interests:** is the most flexible lawful basis for pressing. It is likely to be the most appropriate when using the individual's personal data in ways where there is a minimal privacy impact or where there is compelling justification for the processing of his personal data (ICO 2018).

Documenting and determining the lawful basis before processing personal information is a must for any organization. The privacy policy should reflect the purposes for processing personal data in addition to stating the lawful basis (ICO 2018).

7.1.3 Individual Rights

The regulation enforces a significant amount of individual rights, which are designed to protect and enhance private data. These include:

1. **The right to be informed**: Articles 12, 13 and 14 are responsible for the right to be informed. In a nutshell, individuals have the right to be notified and informed about the collection and use of their personal data; adding an extra layer of transparency. Furthermore, individuals must be provided with information regarding their personal data, including:

   (a) Purposes for processing it

   (b) Its retention period

   (c) Who it is shared with

   Additionally, individuals must be provided privacy information at the time personal data is connected from them, in a concise, intelligible manner and in plain language. Finally, privacy information must be regularly reviewed and updated. By ensuring this right is fully respected, high levels of transparency and trust are built with people (ICO 2018).

2. **The right of access**: it ensures individuals, in Article 15, to "have the right to obtain from the controller confirmation as to whether or not personal data concerning him or her are being processed, and, where that is the case, access to the personal data and the following information:

   (a) The purposes of the processing

   (b) The categories of personal data concerned

(c) The recipients or categories of recipient to whom the personal data have been or will be disclosed, in particular recipients in third countries or international organizations

(d) Where possible, the envisaged period for which the personal data will be stored, or, if not possible, the criteria used to determine that period

(e) The existence of the right to request from the controller rectification or erasure of personal data or restriction of processing of personal data concerning the data subject or to object to such processing

(f) The right to lodge a complaint with a supervisory authority where the personal data are not collected from the data subject, any available information as to their source the existence of automated decision-making, including profiling, referred to in Article 22(1) and (4) and, at least in those cases, meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing for the data subject" (European Union 2016)

In short, individuals have the right to access their data, which can be done by requesting access verbally or in writing with a one month deadline and in most cases data subjects cannot be charged a small fee

3. **The right to rectification**: Article 16 states that "the data subject shall have the right to obtain from the controller without undue delay the rectification of inaccurate personal data concerning him or her. Taking into account the purposes of the processing, the data subject shall have the right to have incomplete personal data completed, including by means of providing a supplementary statement" (European Union 2016).

   In other words, data subject's with this right can request verbally or in writing for inaccurate personal data to be rectified or completed, and this request is accepted or denied within a month (ICO 2018).

4. **The right to erasure**: one of the most important rights in the GDPR is the right to erasure — Article 17. It allows an individual to have personal data erased at his request, either verbally or in writing and with undue delay. However, this right does not always apply. Below is a summary of these special circumstances (European Union 2016):

   (a) Personal data is no longer necessary for the originally collected purposes or otherwise processed

   (b) The data subject withdraws consent, when this is the lawful basis for processing personal data

   (c) The personal data has been unlawfully processed

   (d) In order to comply with legal obligations in Union or Member State law to which the controller is subject

   However, the regulation is also clear where this right **does not** apply (ICO 2018):

   (a) To exercise the right of freedom of information and expression

   (b) In order to comply with legal obligations in Union or Member State law to which the controller is subject

(c) In the exercise of official authority or for the performance of a task carried out in the public interest

(d) When achieving purposes in the public interest, scientific or historical research or statistical

(e) Purposes where erasure is likely to render impossible or seriously impair the achievement of that processing or for the exercise, establishment or defense of legal claims

5. **The right to data portability**: it allows an individual to obtain and reuse their personal data in a structured, commonly-used and machine-readable format (JSON, Extensible Markup Language (XML) or Comma-seperated Values (CSV)) for different purposes (including pseudonymous data). Furthermore, it permits the copy, transfer, and moving of personal data easily and security from one provider to another (European Union 2016).

   This right only applies in the following situations (ICO 2018): The lawful basis for processing personal data is consent or the performance of a contract and this processing is carried out by automated means.

   Additionally, personal data in the scope of the data portability right may include data resulting from observation of an individual's activities, such as (ICO 2018):

   - Search activity in a website or history
   - Location and traffic data or raw data processed by connected objects such as meters and wearable devices

6. Rights in relation to automated decision-making and profiling (not applicable)

7.1.4 Accountability and Governance

As previously seen, accountability is one of the data protection principles which ensures any organization is responsible for complying with the GDPR and therefore must exemplify this compliance.

In order to meet the GDPR accountability requirements, an organization can take several measures such as (ICO 2018):

- Adopting and implementing data protection policies
- Employing the "data protection by design and default" approach
- Maintaining documentation regarding processing activities
- Ensuring the appropriate security measures are implementing
- Recording and reporting personal data breaches, if any
- Carrying out data protection impact assessments for uses of personal data which are likely to result in high risk to the data subject's interests

- Appointing a data protection officer

By being accountable, there will be a growing bond between individuals and any organization, which ultimately builds trust and confidence in it.

Documentation

Under the GDPR, there are explicit guidelines on how to document processing activities, which can help comply with other aspects of the GDPR and improve data governance. For example, there must be records on several aspects relating to processing purposes, data sharing and retention (ICO 2018).
Both controllers and processors have documentation obligations; these records must be kept in writing and up-to-date (ICO 2018).

In small to medium-sized organizations (fewer than 250 employees), only the following processing activities must be documented (ICO 2018):

- Non-occasional
- Could result in a risk to the rights and freedoms of data subjects
- Involves the processing of special categories of data or offense and criminal conviction data

As stated in Article 30 of the GDPR, the following must be documented (ICO 2018):

- The name and contract details of the organization, other controllers (if applicable) and the data protection officer (if applicable)
- The purposes of personal data processing in addition to the description of the categories of individuals and categories of personal data
- Details regarding transfers to third countries with the documentation of transfer mechanism safeguards
- Retention schedules and a description of technical and organizational security measures

Data Protection Officers

Another important duty in GDPR compliance is the appointment of a Data Protection Officer (DPO) in the event an organization is a public authority or if it carries out certain types of processing activities. The duties of DPOs are (ICO 2018):

- Assisting an organization to monitor internal compliance
- Inform and advise on an organization's data protection obligations
- Provide advice regarding Data Protection Impact Assessments (DPIAs)
- Act as a contact point for data subjects in the supervisory authority

Moreover, a DPO must be independent, adequately resourced, an expert in data protection and has therefore to report to the highest management level in an organization. This new

position can be filled by an existing employee or a new hire. However, a DPO must be hired if (ICO 2018):

- The organization is a public authority or body
- The organization's core activities consist of large scale processing of special categories of data

7.1.5 Security, Personal Data Breaches and Penalties

Security is highly encouraged in GDPR, as it requires data processing to be highly secure by means of "appropriate technical and organizational measures", or the "security principle". By doing this, it is required that an organization to have additional requirements regarding the security of processing and they also should consider the costs of implementing these secure measures. For instance, standard secure practices in the GDPR should be the pseudonymization and encryption of data. Moreover, appropriate processes must be in place to test the effectiveness of these measures, with improvements with required improvements, as stated in Article 32(1):

"Taking into account the state of the art, the costs of implementation and the nature, scope, context and purposes of processing as well as the risk of varying likelihood and severity for the rights and freedoms of natural persons, the controller and the processor shall implement appropriate technical and organizational measures to ensure a level of security appropriate to the risk" (European Union 2016)

Security is of extreme importance, as poor security measures leave systems and services at risk with the ultimate downside being to cause real harm and distress to individuals such as (ICO 2018):

- Identity fraud
- Fake credit car transactions
- Targeting of individuals by fraudsters, potentially made more convincing by compromised personal data
- Witnesses put at risk of physical harm or intimidation
- Offenders at risk from vigilantes
- Exposure of the addresses of service personnel, police and prison officers, and those at risk of
- Domestic violence
- Fake applications for tax credits
- Mortgage fraud

The GDPR enforces a strong notion of security which goes beyond cybersecurity. For instance, every aspect of personal data is covered, which bring new responsibilities when handling it — "Confidentiality, integrity and availability" (ICO 2018):

- Personal data must be accessed, disclosed, altered or deleted only by those authorized to do so

- Personal data must be kept accessible, usable, accurate and complete
- Personal data should be recovered in addition to preventing any damage or distress to data subject's involved

Personal Data Breaches

The GDPR covers the scenario where personal breaches occur. These can be both accidental and deliberate, thus ensuring if the unfortunate scenario of a breach occurs, it goes beyond the loss of personal data.

When a personal breach occurs, it must be reported to a supervisory authority such as the ICO in the United Kingdom within 72 hours of becoming aware of the breach, if feasible. Furthermore, in the event a breach occurs and and is likely to result in a high risk of negatively affecting the data subject's rights and freedoms, those affected must be notified without undue delay.

Moreover, the GDPR also enforces the need for any organization to keep a record of any personal data breaches, regardless of the need to notify them or not. The regulation also focuses on the negative consequences for data subjects, as stated in recital 85:

"A personal data breach may, if not addressed in an appropriate and timely manner, result in physical, material or non-material damage to natural persons such as loss of control over their personal data or limitation of their rights, discrimination, identity theft or fraud, financial loss, unauthorized reversal of pseudonymisation, damage to reputation, loss of confidentiality of personal data protected by professional secrecy or any other significant economic or social disadvantage to the natural person concerned" (European Union 2016).

After a breach occurs, the GDPR also has clear and explicit guidelines to what information must be supplied to a supervisory entity (ICO 2018):

- A description of the nature of the personal breach
- The categories and approximate number of data subjects affected
- The categories and approximate number of personal data records concerned
- The contact details and name of the data protection officer (if applicable) or any other contact point where more information can be obtained
- A description of the likely consequences of the breach
- A description of the measures taken (or purposed) to deal with this breach in addition (when appropriate) to measures taken to help mitigate any adverse effects

Futhermore, when reporting about a breach to data subjects, the following information is required to be provided (ICO 2018):

- A description, in clear and plain language, of the nature of the personal breach
- The name and contact details of the data protection officer (if applicable) or any other contact point where more information can be obtained

- A description of the likely consequences of the breach
- A description of the measures taken (or purposed) to deal with this breach in addition (when appropriate) to measures taken to help mitigate any adverse effects

Article 33(5) of the GDPR requires any organization to fully document the facts regarding to breaches, their effects and the mitigation efforts taken.

Penalties

In Articles 83(4) and 83(5), the regulation fixes heavy penalties for possible infringements (European Union 2016):

- For smaller infringements: "fines up to 10 000 000 EUR, or in the case of an undertaking, up to 2% of the total worldwide annual turnover of the preceding financial year, whichever is higher"
- For bigger infringements: "fines up to 20 000 000 EUR, or in the case of an undertaking, up to 4% of the total worldwide annual turnover of the preceding financial year, whichever is higher"

7.1.6 Children

Children, under this regulation, have extra care and protection when personal data is being collected and processed. There must be a lawful basis when processing a child's personal data, where consent is often chosen. However, children aged above 16 and above are able to provide their own consent. In some countries, this age is 13 or older. They also have the same rights are adults over their personal data (ICO 2018).

7.2 GDPR Generic Compliance Plan for DApps

GDPR compliance poses a significant challenge for DApps. The fines for non-compliance are astronomically high, which ensures organizations are committed to obeying it. In this section, a simple guideline to comply, in the best manner possible, to this regulation, despite key issues regarding the design of Blockchains (including Ethereum) — such as immutability —, is presented:

1. Identify the lawful basis for processing personal data. Most of them give individuals the right to erasure, which by default is incompatible with the immutable nature of Blockchains. However, as previously stated, this right does not apply if the lawful basis is only to comply with legal obligations or public tasks
2. If consent is the lawful basis, analyze the parts of the software application that contain mutable data and ensure it is stored in centralized systems; store everything else in one or more smart contracts
3. In one or more smart contracts, store a hash that links both parts (centralized and Ethereum, decentralized)

4. If the user invokes the right to be forgotten, delete his mutable data in the centralized system. This way, the link (hash) between the centralized and decentralized systems is broken and data in the latter might be still accessible albeit it cannot be linked to a specific individual which is outside of the scope of the GDPR, as shown in Figure 40

5. In some cases, personal identifiable information is stored in one or more smart contracts. To comply with the GDPR, one possible approach is to encrypt this personal data, such as performed in DFiles. Note that directly encrypting variables in Ethereum smart contracts is not recommended due to miners having access to all data, including possible private keys used for encryption. This is the main reason DFiles uses the Interplanetary File System

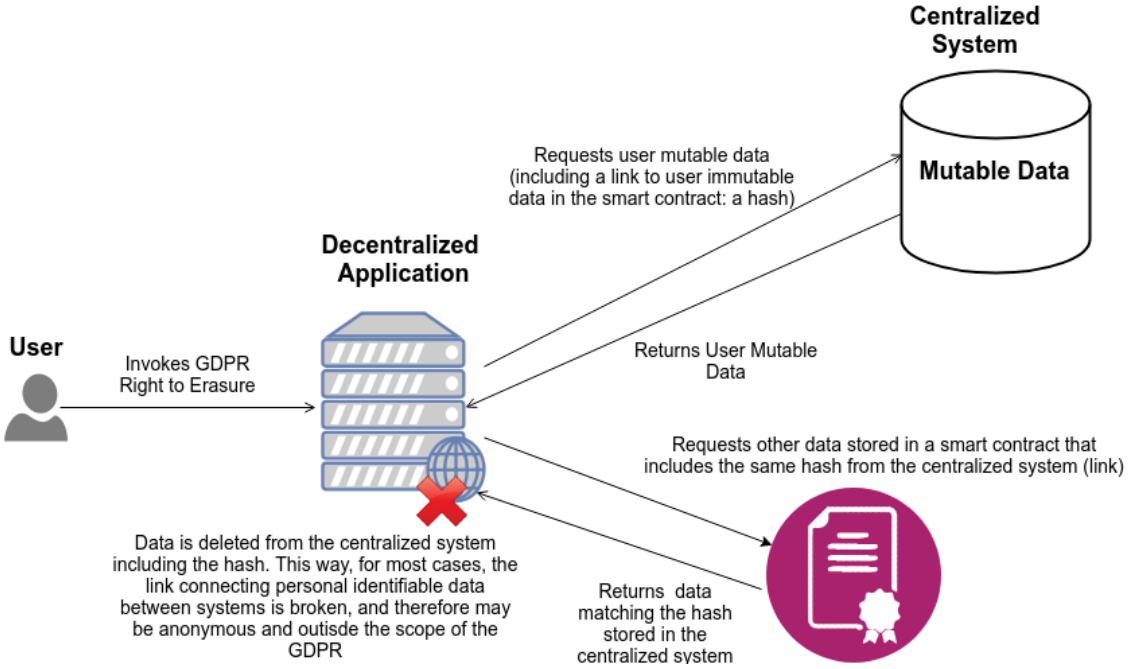

Figure 40: GDPR right to erasure generic compliance for DApps.

7.3 GDPR Compliance Plan for DFiles

In the previous sections, the core aspects of the GDPR were discussed in addition to a generic plan to comply with this regulation for all DApps.
This section attempts to improve the generic plan in presented in the previous section by elaborating a new one specifically for the DFiles DApp, even if some aspects of this regulation might not end up being fully implemented:

- **Data Subjects**: are the users of the DFiles DApp, individuals who have an account in the DApp and use its services (upload, view files, etc).
- **Personal Data**: the first step into creating a GDPR compliance plan is to understand what constitutes personal data. Recall from subsection 6.1.1, the MongoDB user schema. In it, the following information is stored, after the user creates an account and provides consent for its storage:
  - Username

  - Email
  - Password (hashed)
  - Private Key
  - Public Key
  - Ethereum Address

  Username and email, by themselves, may allow a person to be identified, that is, if he or she can be distinguished from other individuals. For example, a username with a person's name might suffice for him to be identified. The same logic applies for the email.
  However, passwords and other generated strings such as hashed passwords, private and public keys and Ethereum Addresses on their own are not enough to fully identify an individual. These can also be considered pseudonimized data, since they cannot be tied to an individual.
  Moreover, if all of the above data is combined, then it can easily identify an individual. Thus, the five collected items from a user are all personal data, and must obey do the specifications of the GDPR.

- **Controller and Processor**: Recall from subsection 7.1.1, the notions of a controller and processor. In other words, a controller determines the purposes of processing personal data. On the other hand, a processor is responsible for processing personal data on the behalf of a controller. The fictional company DFiles Inc is responsible both being the controller and processor for the DFiles DApp. Since the same company is both the controller and processor, there will not be written contract between a processor and a controller, albeit required in the GDPR[3].

  The DFiles Inc company is represented by a fictional employee, Mr.John Smith. He is the face of the company, and responsible for any problems that arise regarding both controllers and processors or about the GDPR in general.

7.3.1 Principles

In this subsection, a plan to comply with the GDPR principles is presented:

- **Lawfulness, fairness and transparency**: in order to comply with this principle, a lawful basis must be identified to collect and process personal data. Consent has been chosen: when a user creates a new account, a clear message asking for user consent to store his data is displayed with a checkbox. In the event the user does not tick the checkbox — does not accept — the user cannot create an account and therefore cannot use any of the features of the DFiles DApp. However, the individual has the ability to withdraw this consent. It will not be considered in DFiles due to the extra layer of complexity added to the DApp.

[3] https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/accountability-and-governance/contracts/

Finally, the user data processed and stored, will **NOT** be for unlawful purposes.
DFiles Inc only handles personal data in ways individuals reasonably expect, it does not mislead people when collecting their data and it considers how the processing may affect individuals.
DFiles Inc is open and honest, and complies with the transparency obligations of the right to be informed.

- **Purpose limitation**: the DFiles DApp complies with this principle, due to the following:
  - The purpose for processing has been identified: **consent** — subsection 7.1.2
  - The purpose above is well documented — subsection 7.1.2
  - The company includes details details about its purposes for processing in its privacy policy for individuals, available at Appendix O. Note that it was written to be clear, simple and precise to be understood by any individual
- **Accuracy**: personal data, when collected, must be accurate. In the DFiles DApp, when users register an account, it is their responsibility to input valid and accurate data. However, personal data such as email addresses are often checked for inaccuracy by sending seldomly emails to users. When invalid data is detected, an email is sent to the user to request updating it to be valid and accurate. Furthermore, the DApp also allows users to edit their personal data at any time with rigorous checks to ensure this data is kept up-to-date and accurate
- **Storage limitation**: as previously explained, the DFiles DApp requires specific data about the user, such as email addresses and their usernames when they create an account. Since uploading files with Ethereum and IPFS requires data from the user's account, it will be stored and collected until he decides to delete his account (right to erasure).
- **Integrity and confidentiality (security)**: there are multiple security measures that protect personal data in place in the DFiles DApp:
  - Password hashing, in the centralized system, when the user creates an account
  - File encryption before uploading to the IPFS
  - When a user requests to have his account deleted, his uploaded files can no longer be accessed
- **Accountability**: this principle requires an organization to take responsibility for what it does with personal data and how it complies with the other principles.

  In order to comply with this principle, DFiles Inc follows a serious os steps (ICO 2018):
  - The company takes responsibility for complying with the GDPR, at the highest management level and throughout the organization
  - The company keeps evidence of the steps taken to comply with the GDPR
  - The company adopts and implements data protection policies
  - The company maintains documentation of its processing activities in addition to implementing appropriate security measures

  - The company records and, where necessary, reports personal data breaches

7.3.2 Individual Rights and Documentation

In this subsection, a plan is created to comply with the GDPR's data subject rights:

- **The right to be informed**: the individual has this right to allow him to understand what kind of personal data is being collected about him. For this reason, a privacy policy document should be created. In DFiles, this is its privacy policy, available at Appendix O and has the following information:
  - The name and contact of the organization
  - The name and contact of the DFiles Inc representative (Mr.John Smith)
  - The purposes of the processing
  - The lawful basis for the processing
  - The retention periods for the personal data
  - The rights available to individuals in respect of the processing
  - The right to withdraw consent
- **The right of access**: to comply with this right, there will be an option in each user's account settings page, to request all the data the DApp has stored about him in the shortest time as possible
- **The right to rectification**: to comply with this right, there will be an edit form in the DFiles user's account settings, where he can change his personal data, in the shortest time possible
- **The right to erasure**: this right may arguably be one of the most important rights an individual has. However, the Ethereum Blockchain is immutable, meaning records cannot be deleted. At a first glance, one might think the Ethereum Blockchain (and others such as Bitcoin) are forever in violation of this right. For this reason, figuring out a plausible way to comply with this right is a must.

  In the DFiles DApp, as discussed earlier, each user-submitted file to the IPFS is encrypted with an unique, automatically generated **private key**. Furthermore, there will be a place in the user's account settings to delete his account, which will only delete his details from the centralized NodeJS/Express server:
  - Email
  - Ethereum Address
  - Username
  - Password
  - Private Key

  When the user requests to delete his account, the private key is deleted too. This means that, any files that were previously decrypted by using the private key, can not

longer be decrypted. In essence, the files are not deleted (unless an IPFS node decides not to serve them anymore), they are fully encrypted and unaccessible since the user deleted the private key along with all his personal information. Thus, data that once could be considered as personal, is truly anonymized and out of reach of the GDPR.

- **The right to data portability**: to comply with this right, the user can request at his account's settings page via pressing a button, to download his data in the machine-readable format, JSON.

Documentation

The GDPR requires any organization to document multiple aspects, including those of the Article 30 (ICO 2018):

- The purposes for processing
- The name and contact details of the organization
- The categories of recipients of personal data
- Retention schedules

The DFiles privacy policy is where all of the above is documented, in addition to more aspects the GDPR requires, in Appendix O.

Last but not least, the GDPR requires any organization to document its processing and controller activities. Two spreadsheet files from the ICO, one[4] for documenting processing activities and another[5] for controller activities will be used.

7.3.3 Children and Data Breaches

The GDPR states several obligations to follow when dealing with children. However, in the DFiles, the minimum age required is 16, with a prompt presented when registering an account to confirm the user is 16 or older.

As for data breaches, organizations must inform a relevant supervisory authority within 72 hours of the breach, when feasible. In the unfortunate event a breach occurs in the DFiles DApp occurs, it will affect the sensitive information that its users stored on the platform. Because of this, the following steps have been taken to prepare for this event:

- A personal data breach is recognized when an attacker compromises the DFiles centralized authentication system, either by:
  - obtaining a copy of the MongoDB database
  - changing user records in the database
  - deleting user records

[4] https://ico.org.uk/media/for-organisations/documents/2172936/gdpr-documentation-processor-template.xlsx

[5] https://ico.org.uk/media/for-organisations/documents/2172937/gdpr-documentation-controller-template.xlsx

- Other possible attacks include decrypting the files uploaded to the IPFS by obtaining the private key from the centralized authentication service

When a breach occurs, Mr.Smith, the face of the DFiles Inc company must be notified, then the affected users are notified via their email addresses, followed by an entry on the data breaches record — Table 33 and Table 34. This record is then provided to the ICO, U.K.'s supervisory authority.

Table 33: GDPR data breach record plan, part one (ICO 2011).

No.	**Details of breach**					
	Date of breach	No. people affected	Nature of breach	Description of breach	How the breach was discovered	Description of data

Table 34: GDPR data breach record plan, part two (ICO 2011).

Consequences of breach	**Measures taken/to be taken**			
	All individuals informed?	Remedial action	Other Regulators informed	When was the ICO notified of the breach?

7.4 GDPR DFiles Implementation

In the previous two sections, the core aspects of the GDPR were discussed in addition to a thorough plan to ensure the DFiles DApp is fully compliant with the regulation.

However, only a very small part of the GDPR was implemented. This includes the **rights to erasure**, **data portability**, **access** and **rectification** and the minimum age of 16 required to use DFiles.
How these were implemented, is discussed and explained throughout this section.

User Registration

When a user creates an account, he is directed to the registration page, where personal data is collected in a form, which is protected by the GDPR. As previously stated, this includes:

- **Username**: the user's username

- **Email**: the user's email
- **Password**: the user's password
- **Ethereum Address**: the user's Ethereum Address

Moreover, a randomly generated RSA pair of public and private keys is also generated and stored. This is used to encrypt the user's files.
Here, a checkbox requires the user to state that he is 16 years of age or older, which ensures a special policy regarding children is not required to be implemented. Figure 41 and Figure 42 illustrate this.

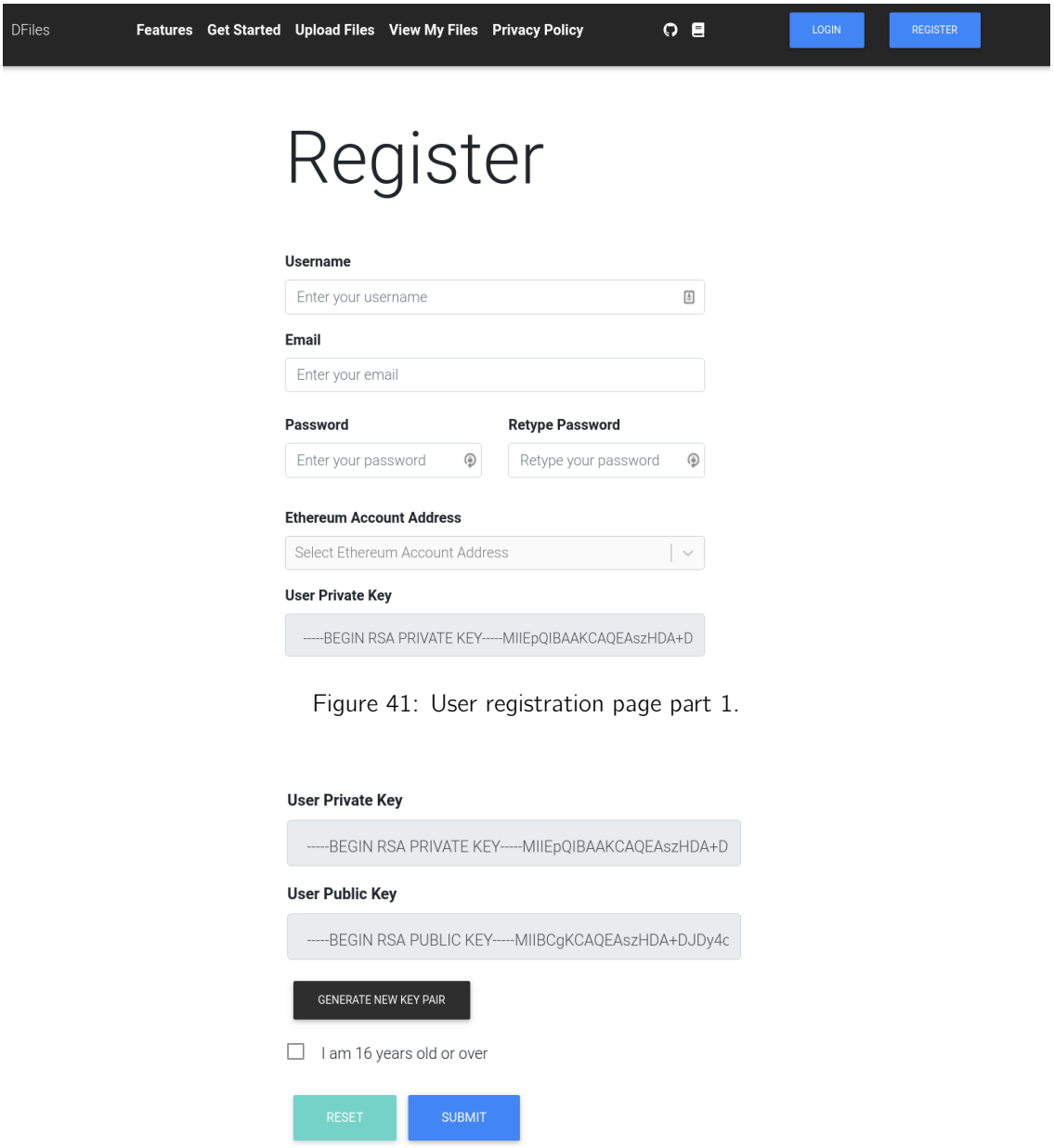

Figure 41: User registration page part 1.

Figure 42: User registration page part 2.

User consent should have been requested when storing and processing his data when agreeing to create his account, as the lawful basis for processing. Unfortunately, due to the massive

scale of the GDPR and time contraints, this was not possible to be implemented. For the rest of this section, it is assumed the user voluntarily gave his consent for DFiles, Inc to process his personal data. As a consequence, the ability for the data subject to withdraw consent is not possible.

Right to Access

After successfully creating an account with the DFiles DApp, each user has a specific account settings page, where each of the previously mentioned rights can be invoked. This page is displayed in Figure 43 and displays most of the individual's personal data.

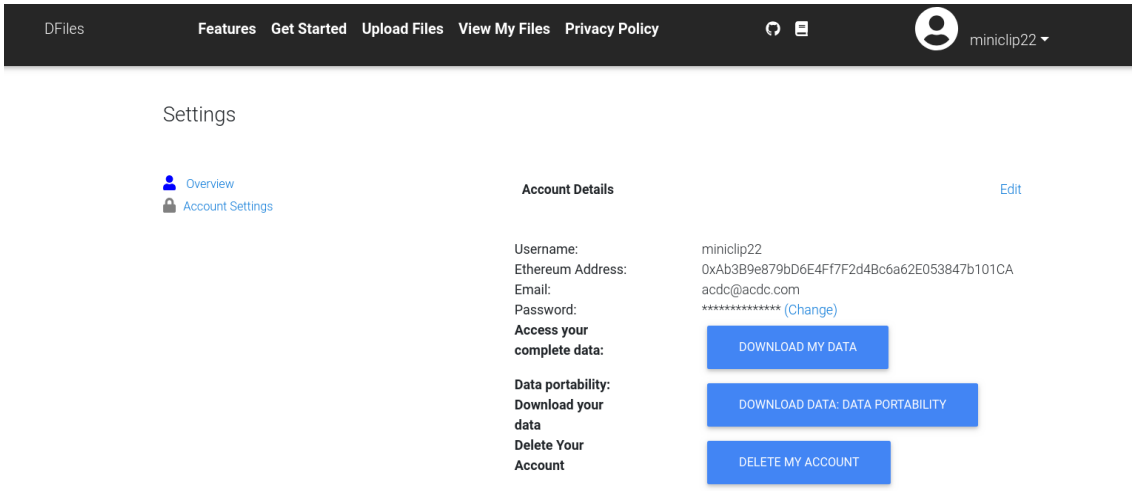

Figure 43: User account settings page.

The right to access can be invoked by clicking the "**DOWNLOAD MY DATA**" button. When doing so, all information about a specific user that DFiles Inc has about him can be, almost instantly, downloaded. This includes all the files he uploaded and its metadata, his personal data (as mentioned above), IPFS URL and hash of all files uploaded. When this button is clicked, the user is prompted to download a JSON file containing this data, as detailed in Figure 44.

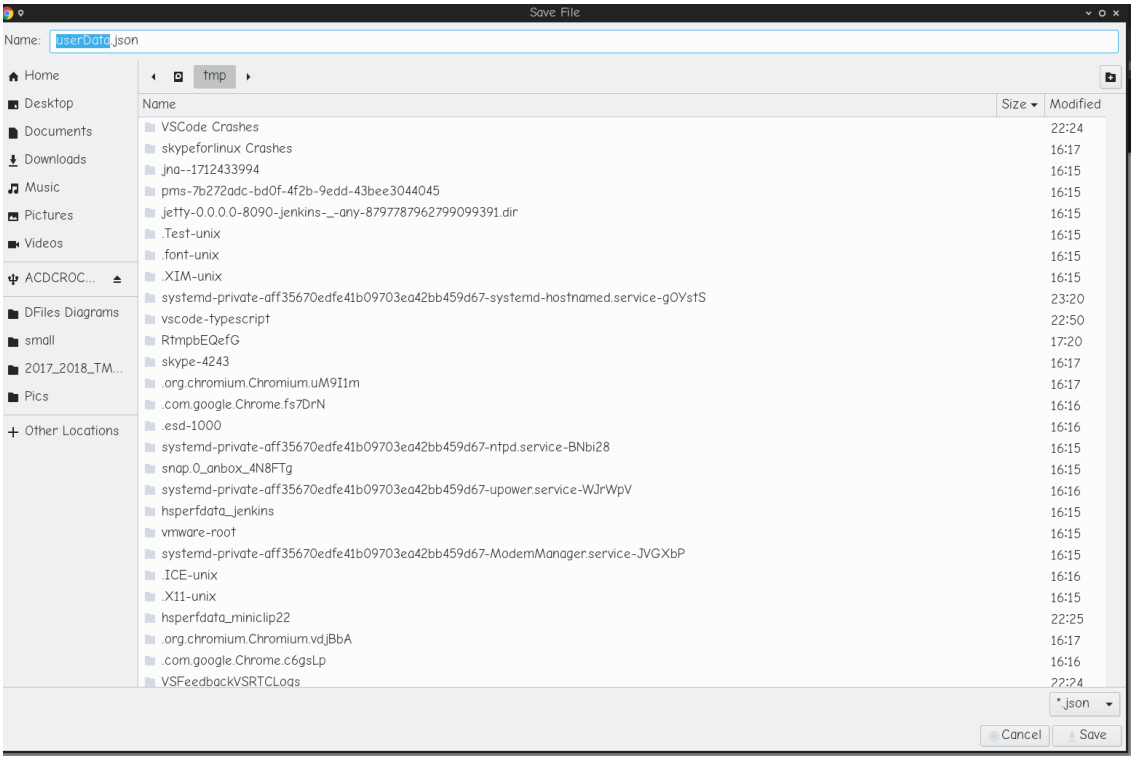

Figure 44: Right to access JSON file download prompt.

Once the user opens this file, it contains the data mentioned above, in JSON, as shown in Figure 45.

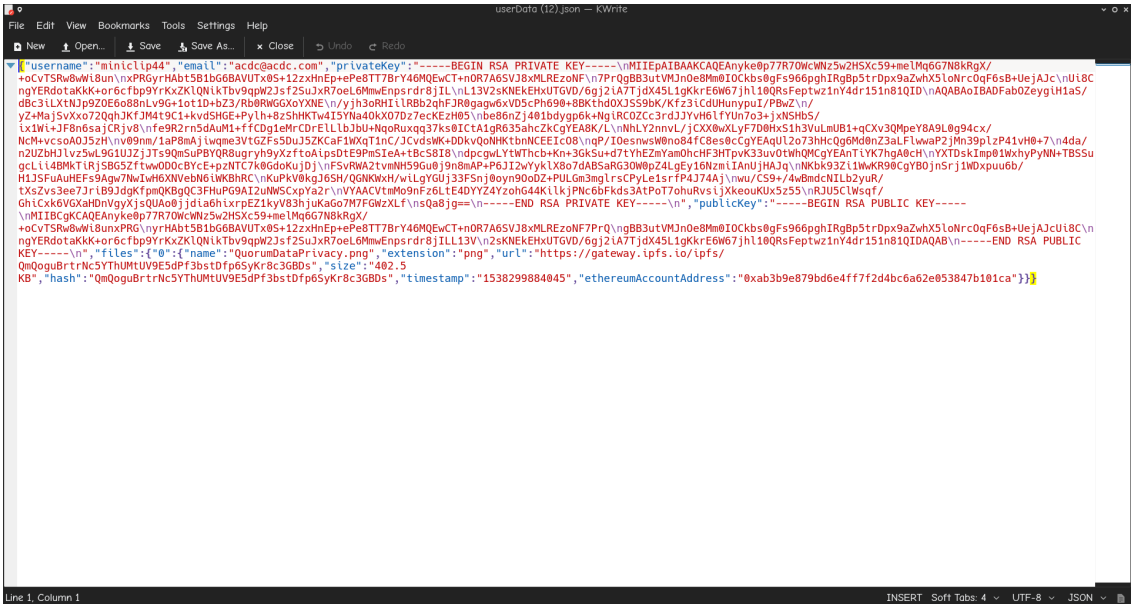

Figure 45: The downloaded JSON file containing all the data DFiles Inc has on a specific individual.

Right to Data Portability

The right to data portability allows a user to download his personal data, in a secure way, in a machine-readable format for usage in other products or services. The key difference from the right to access is the volume of data included. In the right to data portability, only the individual's personal data can be downloaded, whereas in the right to access this is **ALL** of his personal data.

Similarly to the right to access, the right to data portability can be invoked by clicking in the **DOWNLOAD DATA: DATA PORTABILITY** button, as displayed in Figure 43. Once this button is clicked, a similar prompt to Figure 44 is displayed. Finally, the individual can then view his downloaded personal data: Figure 46.

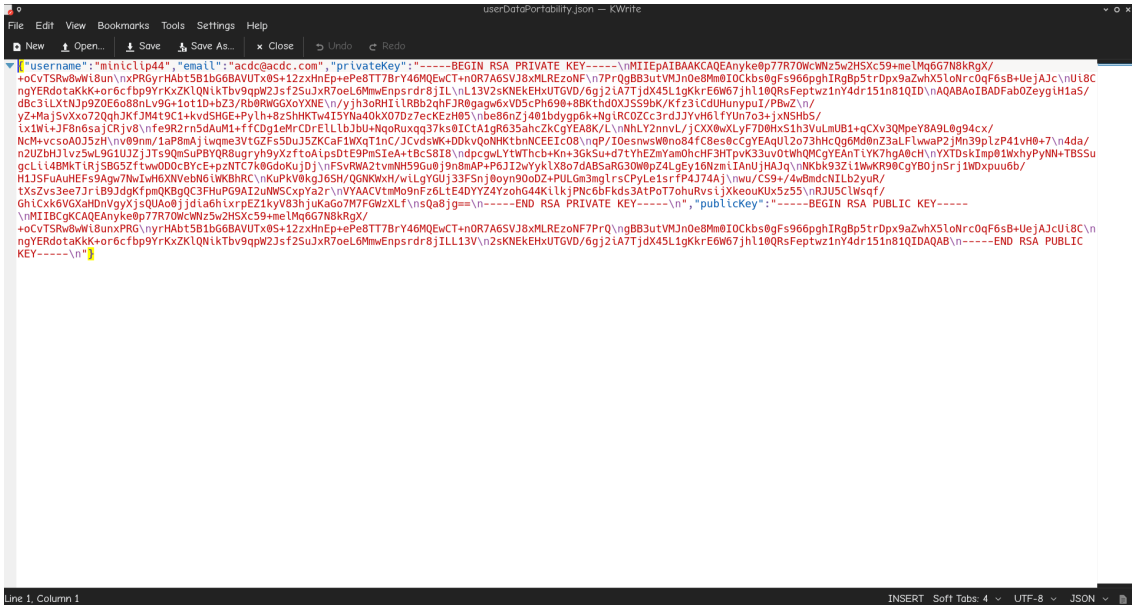

Figure 46: The downloaded JSON file containing **only** the individual's personal data.

Right to Rectification

The right to rectification allows the individual to promptly change invalid or missing personal information. In the DFiles account settings page (Figure 43), the data subject is presented with an **Edit** link or **Account Settings** (on the upper left), which redirects to the account edit page, as seen in Figure 47.

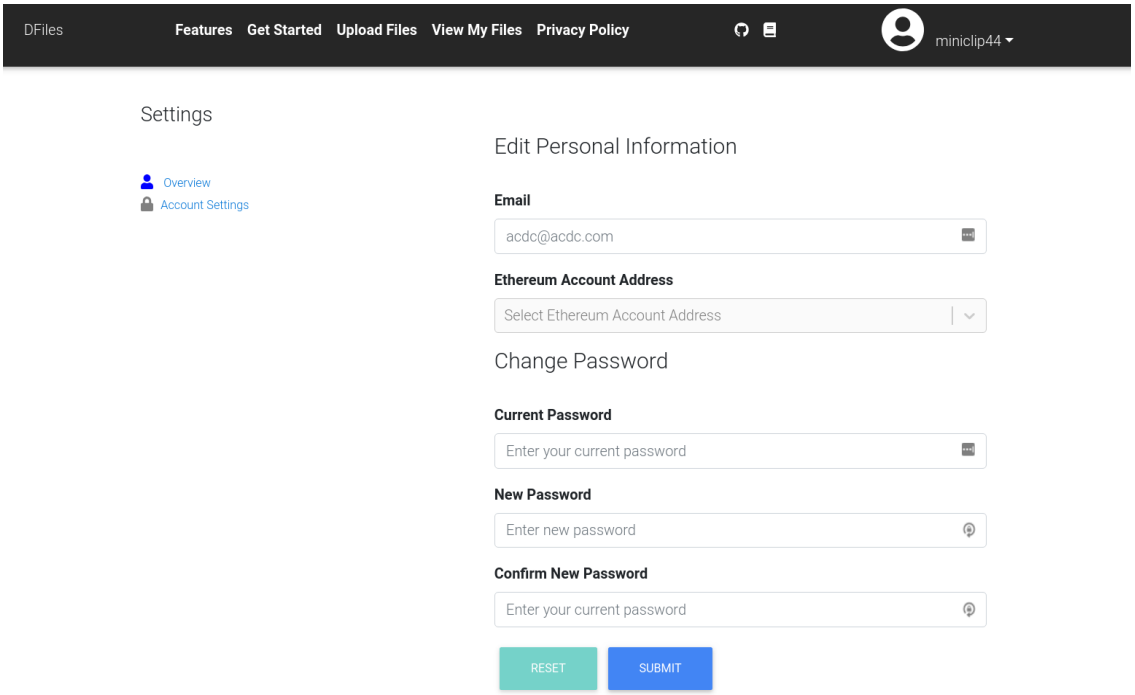

Figure 47: The account settings page for an individual to edit or add missing information.

Right to Erasure

The right to erasure is the most difficult GDPR right to implement, as it states that an individual can request that his personal data be deleted. By recalling the immutable nature of the Ethereum Blockchain, this right appears to be impossible to comply. In DFiles, recall that the DApp has a centralized and a decentralized component. In the centralized one, the individual's personal data is stored, in addition to which Ethereum address he currently has. In the DFiles core smart contract, this address is also stored, which acts as a link between the centralized and decentralized parts, in addition to the IPFS hash and uploaded file metadata.

When the user invokes this right, the link (Ethereum address) between the centralized and decentralized systems is broken, in addition to the deletion of his private and public keys used to encrypt files. Technically, the encrypted files are available for long periods of time in an IPFS node, though they cannot be tied to any specific data subject and thus are outside of the scope of the GDPR (anonymous data). Figure 48 illustrates this idea.

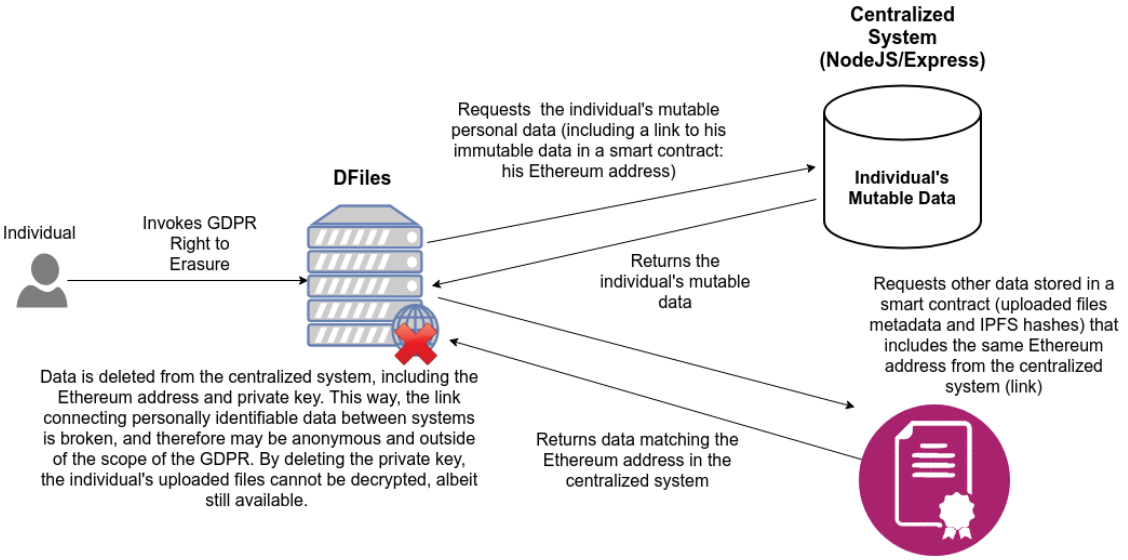

Figure 48: The overall process of an individual invoking his right to erasure.

To demonstrate this, an individual with an account in the DFiles DApp uploads a new file to the IPFS, as seen in Figure 49.

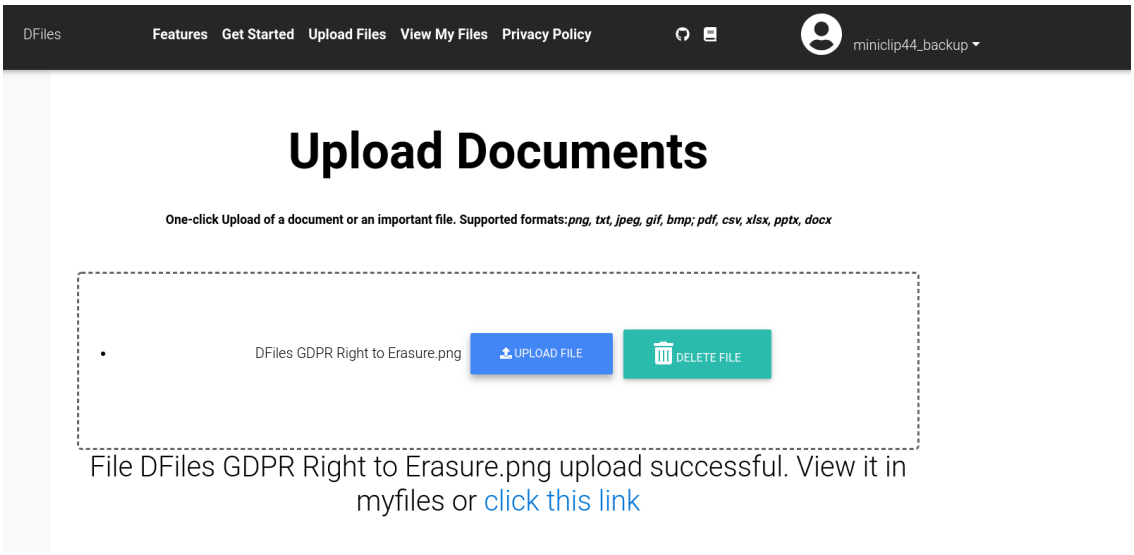

Figure 49: The individual uploading a file to DFiles.

A URL to the encrypted file is then displayed. When the individual clicks this link, the encrypted file (with his private key) is presented as a series of strings of text, Figure 50.

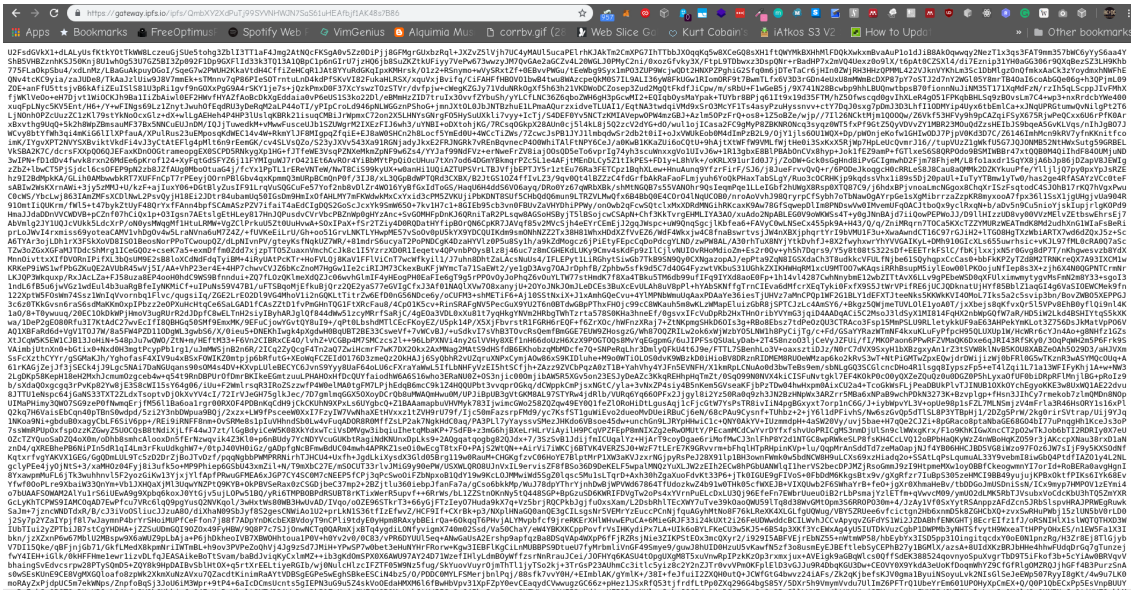

Figure 50: The individual's encrypted file.

The data subject then clicks the **My Files** sidebar option. A list of his encrypted uploaded files is then made available. If the user individual clicks the **DECRYPT AND DOWNLOAD** button, the selected file is then decrypted and made available for download: Figure 51 and Figure 52.

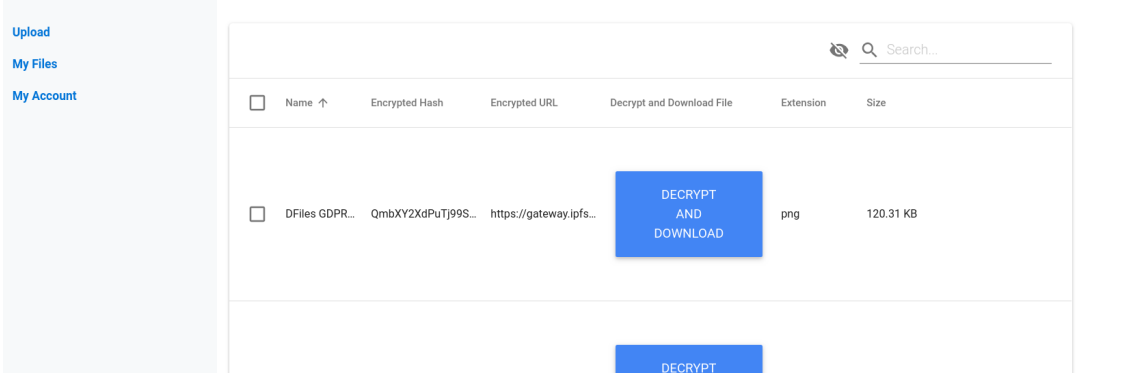

Figure 51: The individual's encrypted uploaded files list.

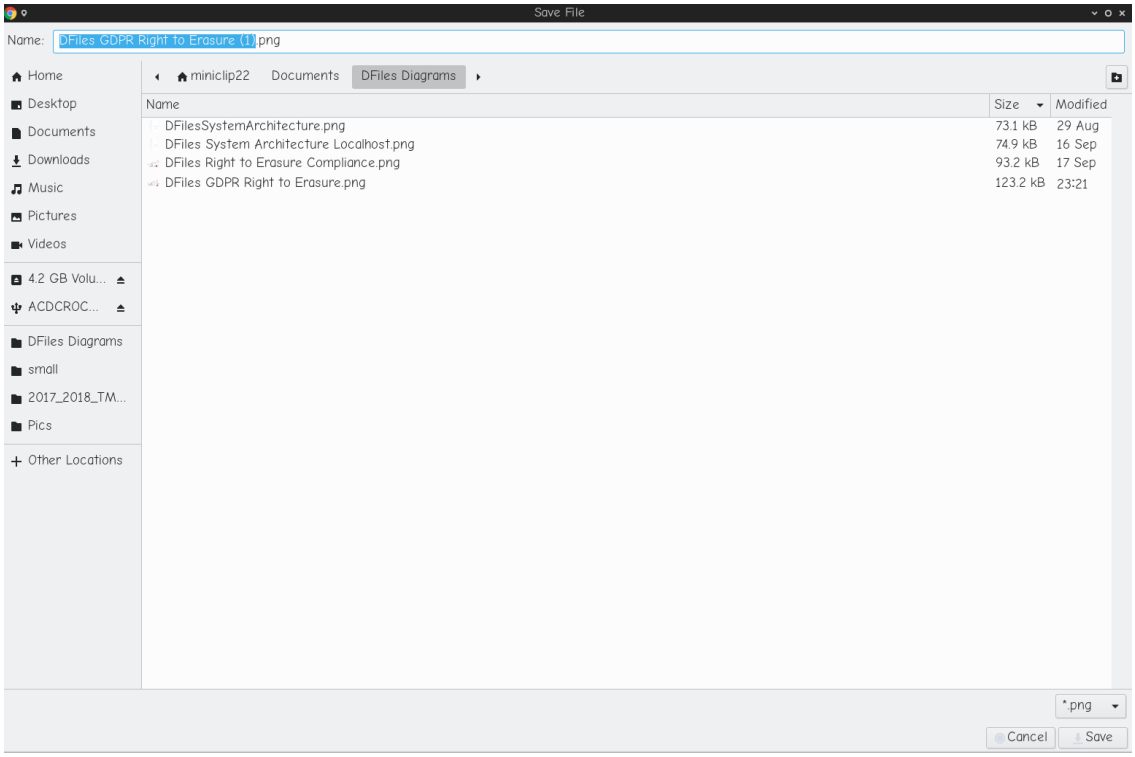

Figure 52: The decrypted file download prompt.

After this, the individual then proceeds to delete his account, by accessing the **account settings** page and selecting the download button, as shown in Figure 43. A message should appear that his account was deleted, as displayed in Figure 53.

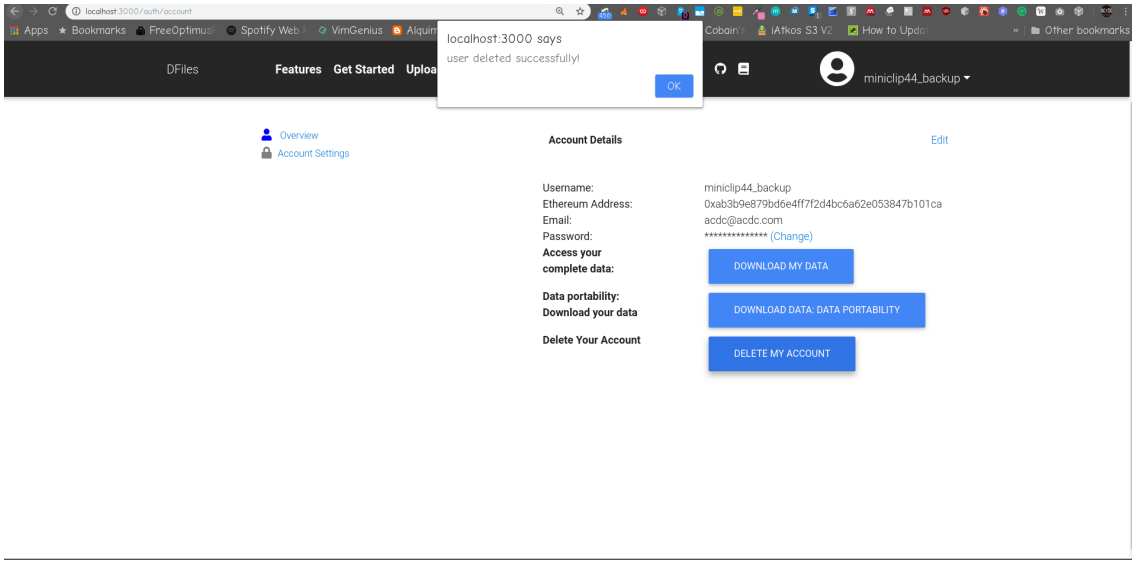

Figure 53: A message confirming the individual deleted his account.

All of the individual's uploaded files are forever encrypted (or at least until the IPFS nodes decide to store them), anonymized and outside the scope of the GDPR. This way, the right to rectification is complied albeit with immutable data.

7.5 GDPR Key Changes

The GDPR aims to improve from the previous 1995 directive. The world, from that point until today, is understandably different. Today, the world is increasingly data-driven and requires legislation to accurately protect every individual's personal data in the EU. However, the GDPR will holds true to some principles first designed in 1995.
In this section, it is discussed the multiple key changes of the GDPR from the 1995 directive.

Increased Territorial Scope

In the 1995 Data Protection Directive, its territorial applicability was ambiguously referred to data process in 'in context of an establishment' (EU GDPR 2018).

The GDPR makes its territorial scope explicitly clear: it "will apply to the processing of personal data by controllers and processors in the EU, regardless of whether the processing takes place in the EU or not. The GDPR will also apply to the processing of personal data of data subjects in the EU by a controller or processor not established in the EU, where the activities relate to: offering goods or services to EU citizens (irrespective of whether payment is required) and the monitoring of behavior that takes place within the EU. Non-Eu businesses processing the data of EU citizens will also have to appoint a representative in the EU" (EU GDPR 2018).

Penalties, Consent, Data Subject Rights and Other Key Changes

Arguably one of the biggest improvements from the 1995 directive is the penalties policy in the GDPR. In the 1995 legislation, companies could provide long illegible terms and conditions filled with legal terms that few would understand as a request for user consent. As already mentioned in subsection 7.1.2, the GDPR obliges companies to request consent in an intelligible, easy accessible form and distinguishable from other matters, with the purpose for its processing attached. Also, it must be as easy to withdraw consent as to provide it (EU GDPR 2018).

As previously discussed, the GDPR introduces several individual rights:

- Right to access
- Right to erasure
- Right to data portability

Moreover, comparative to the 1995 act, it also adds the data protection by design and default principles.
As for other key chances, the GDPR introduces a new policy for breach notification and also data protection officers.

Chapter 8

Evaluation

When developing software, there must be some form of evaluating if, for a specific timeframe, this software reaches the proposed objectives and goals. To do this, there are multiple ways ranging from project planning and discussion, to a comprehensive statistical analysis. Recall from chapter 1 where the main objectives for this work are discussed:

- Can DApps be fully compliant with the GDPR?
- Is it feasible to protect user data by encrypting it, in Ethereum (Homestead)?

Also recall, from section 6.1, the two system architectures the DFiles DApp has: a local one, with Ganache and a local IPFS node, and a deployed one, with an Infura IPFS and Rinkeby Ethereum nodes.

In this chapter, a statistical analysis is performed (this uses the local system architecture: Ganache and a local IPFS node) to understand if encrypting personal data (or files) is feasible or not before sending them to the IPFS node. The ultimate aim is to understand if DApps can comply with the right to erasure (as this is the main conflicting point between Ethereum DApps and the GDPR), by encrypting files with the IPFS.

8.1 Statistical Analysis

Recall from chapter 4 the concepts of **gas limit**, **gas used by transaction**, **gas price** and **total transaction cost** as these are important aspects throughout this section.

Two versions of the DFiles DApp were developed: one without user authentication and file encryption, and another with both user authentication and file encryption/decryption in two separate Git branches.

First and foremost, random files were selected with various extensions: .pdf, .docx, .xlsx and .pptx. They were then divided into four different groups: **small** (1KB-1MB), **medium** (1MB-20MB), **large** (20MB-200MB) and **extra-large** (200MB-2GB). For the same files in both versions, the elapsed time of when the user clicks the "upload file" button, the file gets sent to the local IPFS node (encrypted or unencrypted) and the user clicking the button to accept the transaction is measured in addition to the total transaction cost in Ether (provided by metamask and varies depending on the gas limit and gas price). Appendices

K and L show some of this collected data. However, due to hardware limitations with file encryption, only files up to 14.2 MB were considered for the following statistical analysis. The total transaction cost and upload elapsed times were still recorded for comparison in the overall conclusion between files with and without encryption.

8.1.1 Small Files

Appendix M shows the descriptive statistics for unencrypted and encrypted files as well as the test results for their comparison.

- Amount + Gas fee (Total) — the distributions are right-skewed for both unencrypted and encrypted files, i.e., most files exhibit small values (Fisher's skewness coefficient is positive in both file groups). The average amount is 0.0004024 ETH and 0.0003885 ETH for unencrypted and encrypted files respectively. The minimum value is 0.0003750 ETH and 0.0003640 ETH respectively, the first quartile is 0.0003760 ETH and 0.0003640 ETH respectively, the median is 0.0003770 ETH and 0.0003650 ETH respectively, the third quartile is 0.0004380 ETH and 0.0004260 ETH respectively and the maximum is 0.0004720 ETH and 0.0004600 ETH respectively. Therefore, it can be concluded that files with small values are predominant in both groups. Such concentration leads to very small variability as shown by the coefficient of variation (8.4% and 8.5% respectively).In order to compare the distributions of both groups, the two samples were first tested for normality with the Shapiro-Wilk test (5% significance level). Normality is strongly rejected in both groups (p-value < 0.001). Therefore, a comparison of both samples was based on the Wilcoxon test (paired samples) whose p-value is 0.007. Therefore, the average amount + gas cost is larger for unencrypted files. Figure 54 clearly shows that conclusion
- Uploaded elapsed time — the distributions are slightly left-skewed for both unencrypted and encrypted files, i.e., most files exhibit moderate and large times (Fisher's skewness coefficient is negative in both file groups). The average time is 3.6 seconds and 4.3 seconds for unencrypted and encrypted files respectively. The minimum value is 2 seconds and 3 seconds respectively, the first quartile is 3 seconds and 4 seconds respectively, the median is 4 seconds for both groups, the third quartile is 4 seconds and 5 seconds respectively and the maximum is 5 seconds for both groups. Therefore, it can be concluded that files with moderate and large times are predominant in both groups. Such concentration leads to a small variability as shown by the coefficient of variation (25.7% and 15.8% respectively)
- The results of the Shapiro-Wilk test show that normality is strongly rejected in both groups (p-value of 0.003 and 0.0001 respectively). The Wilcoxon test has a p-value of 0.001 and it can be concluded that time is larger for encrypted files. Figure 54 clearly shows that conclusion

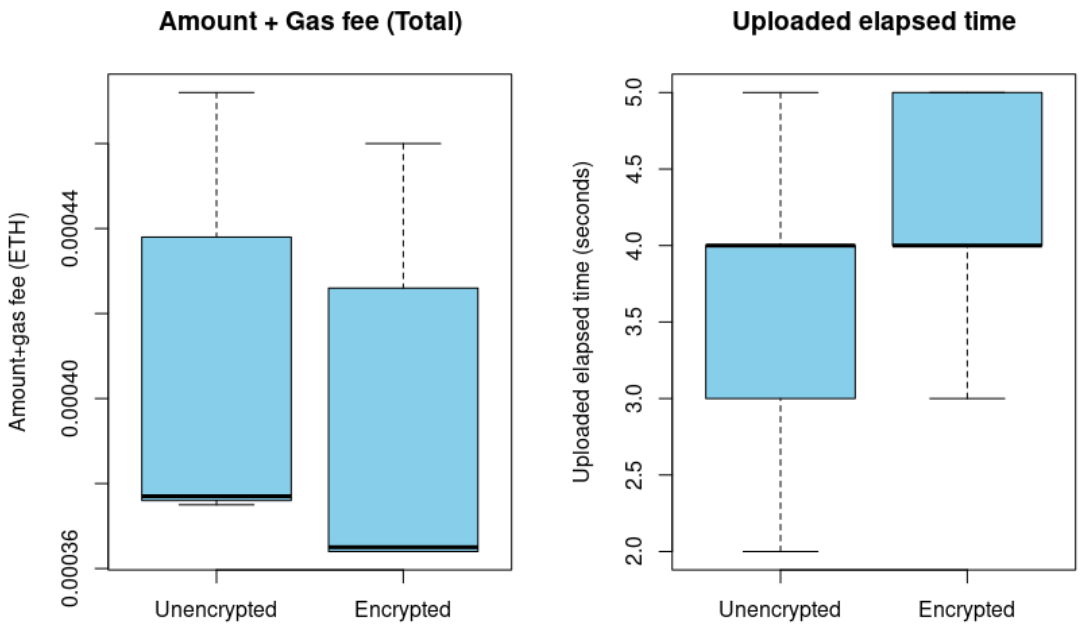

Figure 54: Boxplots for small files.

8.1.2 Medium Files

Appendix N shows the descriptive statistics for unencrypted and encrypted files as well as the test results for their comparison.

- Amount + Gas fee (Total) — most unencrypted and encrypted files are either large or small. The average amount is 0.0004087 ETH and 0.0003938 ETH for unencrypted and encrypted files respectively. The minimum value is 0.0003750 ETH and 0.0003630 ETH respectively, the first quartile is 0.0003753 ETH and 0.0003640 ETH respectively, the median is 0.0004225 ETH and 0.0003805 ETH respectively, the third quartile is 0.0004380 ETH and 0.0004268 ETH respectively and the maximum is 0.0004400 ETH and 0.0004280 ETH respectively. Variability is very small as shown by the coefficient of variation (7.7% and 7.9% respectively)
- The results of the Shapiro-Wilk test show that normality is strongly rejected in both groups (p-value < 0.001). The Wilcoxon test has a p-value of 0.0002 and it can be concluded that the average amount is larger for unencrypted files. Figure 55 shows that conclusion
- Uploaded elapsed time –the distributions are right-skewed for both unencrypted and encrypted files, i.e., most files exhibit small and moderate times (Fisher's skewness coefficient is positive in both file groups). The average time is 6.3 seconds and 29.6 seconds for unencrypted and encrypted files respectively. The minimum value is 5 seconds and 6 seconds respectively, the first quartile is 6 seconds and 10 seconds respectively, the median is 6 seconds and 19 seconds respectively, the third quartile is 7 seconds and 50 seconds respectively and the maximum is 9 seconds and 69 seconds respectively. Therefore, it can be concluded that files with small and moderate times

are predominant in both groups. However, variability is small for unencrypted files and is large for encrypted files as shown by the coefficient of variation (18.8% and 82.2% respectively)

- The results of the Shapiro-Wilk test show that normality is strongly rejected in both groups (p-value of 0.019 and 0.002 respectively). The Wilcoxon test has a p-value of 0.0002 and we conclude that the average time is larger for encrypted files. Figure 55 clearly shows that conclusion
- Size — the distribution is right-skewed, i.e., most files exhibit small and moderate sizes (Fisher's skewness coefficient is positive). The average size is 6.5 MB. The minimum size is 1.2 MB, the first quartile is 3 MB, the median is 5 MB, the third quartile is 12 MB and the maximum is 14.2 MB. Therefore, it can be concluded that files with small and moderate sizes are predominant,but some files have a much larger size which leads to high variability as shown by the coefficient of variation (72.3%)

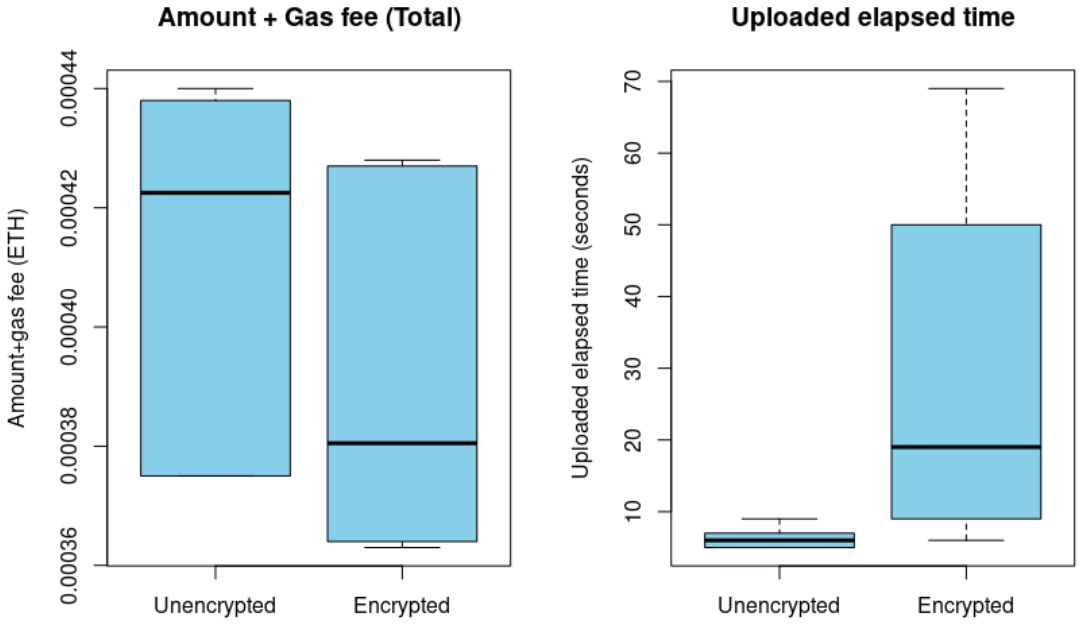

Figure 55: Boxplots for medium files.

8.2 Evaluation Conclusion

The statistical analysis performed on small to medium files up to 14.2MB show the average total transaction cost is slightly larger in unencrypted files. However, as expected, the average upload time was bigger for encrypted files. As for medium ones, the average total transaction cost is again slightly larger for unencrypted files and the average upload time is larger in encrypted files.

In addition to this, if one compares the last recorded values for encrypted and unencrypted values, from all categories (small, medium, large and extra-large) in appendices A and B, it clearly demonstrates that the last 2 file encryption times for files up to 800 MB, are less than the upload elapsed time for encrypted files up to 14.2 MB.

The conclusion is that encrypting files to comply with the right to erasure is a valuable option for small to medium files up to 14.2 MB. From there, without considering hardware encryption limitations, upload times tend to grow exponentially. For this reason, encrypting medium to extra-large files is not feasible as of late 2018. Ethereum and the IPFS must advance to allow better privacy techniques with newer versions which may allow full GDPR compliance.

Chapter 9

Conclusions

This chapter finalizes the main part of this document. The most important aspects of this work are described in chapter 5, chapter 6 and chapter 7, followed by its evaluation in chapter 8.
In the remaining of this chapter, this dissertation's work is summarized in an appropriate section entitled "Work Summary", in addition to its contributions and its limitations and future work. Finally, a personal overview.

9.1 Work Summary

This dissertation aims to achieve complete GDPR compliance in the Ethereum Blockchain with its DApps, by attempting to solve the issue of its immutability and the regulation's right to erasure. To do this, a deep knowledge was required about Ethereum, its smart contracts and other decentralized technologies such as the IPFS. In an early phase, the multiple Blockchains and cryptocurrencies were explored with the objective of understanding if data protection is in fact possible in Ethereum by analyzing and comparing other Blockchains to Ethereum. Among others, the end result of this research was the Quorum private Blockchain, which runs on top of Ethereum albeit aimed at private organizations.

After this stage, a deep research was carried into how smart contracts work and interact with each other in addition to exploring the concepts relating to Ethereum transaction costs and gas. As a direct result of this study, a more clear picture was formed on how to explore Ethereum and its smart contracts to build a data protection prototype which aims at achieving full GDPR compliance. To do this, another intense round of studying this regulation was required. When this started, heavy penalties for not complying with the GDPR were still not in effect. From May 25th, 2018 onwards, these severe penalties were now being enforced. The main challenge here was the lack of knowledge regarding overall legislation such as the GDPR, which proved to be a cumbersome phase in this dissertation's work.

Once most of the GDPR concepts were fully understood in addition to Ethereum, its problems and its smart contracts, then the idea surged to explore file upload and download while encrypting them to protect user data. This way, data could be collected to be studied and asserted if encryption is a valid alternative to comply with the right to erasure. The GDPR has a specific article where it clearly states that anonymous personal data is not covered by it. The objective is simple: encrypt and decrypt user uploaded files; once he invokes his right to erasure, the encryption keys are deleted and therefore data is still present in

the Ethereum Blockchain albeit completely anonymous and encrypted; thus, outside of the GDPR's scope.

Another difficult challenge was to figure out the best way to store data in smart contracts. Directly encrypting variables in them and/or attempt to store whole files represents an enormous transaction cost that cannot be currently paid. For this reason, there was another learning stage to understand if there are other decentralized techniques to store files. Two of the most promising are IPFS and Whisper. The former was chosen due to the simplicity and popularity of this file system.

After combining all the learnt knowledge from Ethereum, its smart contracts, and IPFS, this dissertation case study was then designed: DFiles, a DApp which attempts to copy the Dropbox file sharing service, albeit with decentralized technologies. During early stages of this work, it was quickly realized that full decentralization and GDPR compliance was impossible due to the fact of not having a method of deleting at least part of the user's personal data. To resolve this issue, a centralized component in the DApp was created with the aim of storing mutable personal data.

At this stage, there were multiple challenges with the frontend side of the DFiles DApp, as ReactJS is difficult to begin working with. However, once these problems were sorted out, the end result was a beautiful DApp using modern technologies. DFiles also aims to adhere to Blockchain Software Engineering, where the software development lifecycle was adapted to include new methodologies related specifically to Blockchain software development such as a primary focus on testing. Of all the phases in the software development lifecycle, the deployment is certainly the hardest one, as deployed smart contracts must be ensured they are 100% bug free, secure and thoroughly tested. Once this process begins, the code cannot be changed. Furthermore, there is also the problem of ensuring the overall transaction costs remain affordable, as miners can reject this process if the overall gas cost is too low or too high. Here, two different methods of deployment were also discussed: one with a local Rinkeby Ethereum test network node, and another by using Infura's Rinkeby Ethereum node. Both of these processes have advantages and disadvantages. Infura's node was chosen for deployment, as it is easier, more convenient and does not require a substantial amount of disk space and syncing time. Also during deployment, there was a bug in one of the DFiles smart contract which caused the overall DApp to behave unexpectedly. To solve this issue, multiple deployments had to be made, with an estimated total deployment cost being around 8ETH, or 1,820.80 USD.

Once DFiles was developed, with an Agile methodology and by iterations, it had to be evaluated regarding the feasibility of GDPR compliance. To do this, first data was collected about the user upload times for both encrypted and unencrypted files, in addition to their overall transaction costs. After this, a statistical analysis was performed with this data to conclude if in fact, by encrypting personal data in the IPFS, full GDPR compliance can be achieved by respecting the right to erasure. It found this is only feasible up until files with 14.2MB of size. In today's world, this file size is unacceptable, as it is simply too small in many causes to have personal data stored and encrypted.

In conclusion, Ethereum, its smart contracts and the IPFS are still a long way to achieve mainstream usage and maturity. As of late 2018, Ethereum's most recent version is Homestead, which does not, by itself, have any way of storing private information or transactions in it. As it can be seen currently, there is only one option for DApp development in order comply to some extent with the GDPR, with another one reflecting the current state of DApp development and data protection regulations such as the GDPR:

1. Use file or other form of feasible encryption for small amounts of data, and store everything else offchain (i.e. in centralized solutions)
2. Do not develop DApps, and wait until the European Union decides to draft and pass specific regulation for general Blockchain applications, which includes Ethereum DApps.

9.2 Contributions

This dissertation's work is highly important, relevant and complex. To begin with, it aims to tackle the difficult issue of achieving GDPR compliance in DApps. When it started (October 2017), this regulation was still not enforcing penalties to organizations; these at the time did not understand how they could comply with it despite having traditional, centralized software solutions. This work mixes two highly difficult areas: privacy and Ethereum Decentralized Applications development. Both presented a substantial amount of problems throughout this year. To fully understand the magnitude of this work, the following can be described as its contributions to society:

1. Tackling data protection and its subset user privacy in Ethereum Blockchain Decentralized Applications
2. Understanding how can DApps comply to a certain degree with the GDPR, despite the issue of the right to erasure, which appears to contradict with the regulation
3. Being one of the few DApps (DFiles) to fully embrace Blockchain Software Engineering
4. Discussing the general approach to achieve GDPR compliance in standard DApps
5. Documenting both DFiles GDPR compliance and implementation
6. Evaluating if data protection and privacy in DFiles is feasible in addition to if it is possible to comply with the GDPR's right to erasure

Last but not least, most of this work resulted in two book chapters:

1. "Blockchain: Past, Present and Future"[1], submitted for publication and it is currently being reviewed. A two-page abstract was previously accepted
2. "Privacy with Ethereum Smart Contracts"[2], submitted for publication and is pending review from the editors

[1] https://www.igi-global.com/publish/call-for-papers/call-details/3126

[2] https://mragnedda.wordpress.com/2018/03/03/call-for-chapters-blockchain-and-web-3-0-social-economic-and-technological-challenges/

9.3 Limitations and Future Work

Ethereum is clearly an exciting and evolving technology. However, there are several limitations that had a major impact in this work. For instance, the severe gas cost of having smart contracts with loops discouraged its adoption in the DFiles core smart contract. Moreover, the lack of support for JSON objects is also a major aspect to take into consideration, as smart contracts in Solidity could potentially be more efficient if instead of using a structure, a standard JSON object is used. On top of this, the IPFS clearly has a long way to go for DApps to become mainstream.

To begin with, it has a growing problem of storing very large files as demonstrated by DFiles. If encryption is required, then the maximum file size possible is substantially lower. Ethereum, IPFS, and data protection techniques will evolve significantly in the future, thus shining a bright light for future DApps that require data privacy and protection.

Previously, it was mentioned that the IPFS is immutable and files uploaded to nodes running this file system are stored probably forever. However, these nodes have the option to delete all files from their system, if they please.
Another important limitation in the Interplanetary File System is the fact that there is not a financial incentive put in place to ensure IPFS nodes store the user's files for a considerable time in the future. Today, IPFS nodes such as Infura typically delete uploaded files after a short period of time and heavily limit uploaded files to their node by size. A solution to this is to operate a local IPFS node, — where all files are kept for as long as required — but does not adhere to the goal of achieving full decentralization and has the limitations already mentioned above.

As for future work in DFiles, there is an impressive array of improvements for this DApp in both frontend and backend, as only a prototype was developed. In the frontend, these include:

1. Better documentation for all frontend code
2. Usage of ReactJS patterns such as Higher Order Components (HOCs)
3. ReactJS state definition and handling with Redux[3], a library used to efficiently manage in ReactJS web applications
4. Unit testing with Facebook Jest[4]
5. Better display of loading screens to the user with CSS animations

As for the backend, there are also several improvements to be made:

1. Deployment of the centralized component in DFiles: nodeJS/Express server and the MongoDB database
2. Mobile client with **Trust Wallet App**[5] instead of metamask to connect to the Ethereum Rinkeby test network
3. Implementation of a share uploaded files feature

[3] https://redux.js.org/
[4] https://jestjs.io/
[5] https://trustwalletapp.com/

4. Creation of a DFiles Ethereum token, which is used for users to pay for their uploaded files instead of standard Ether
5. Migration from the HyperText Transfer Protocol (HTTP) to the HyperText Transfer Protocol Secure (HTTPS) in the overall DApp
6. Development of a better Login system with **OAuth**[6]
7. Creation of a metamask not found/invalid Ethereum address web page

In addition to the backend and frontend improvements mentioned above, there is a substantial amount of work to be done to achieve complete General Data Protection Regulation compliance. Recall from chapter 7, that only a small subset of this regulation was in fact, implemented albeit planned. For this reason, the following is a list of additional tasks to be done sometime in the future in order to achieve this goal:

1. When the lawful basis is consent, allow the individual to withdraw it at any given time
2. Develop clear forms and messages to better gather user consent
3. Implement the full extent of the **data protection by design and default** principles

9.4 Personal Overview

This year has been the most difficult of my life, as this dissertation involved all my dedication and effort to fully embrace Blockchain technology and its data protection and privacy problems. However, I have gained invaluable knowledge in developing standard DApps with DFiles, while adhering to Blockchain Software Engineering.

As for personal challenges, I had so many that I lost count. To begin with, understanding the core concepts of Blockchain technology in general, took me several months to do so. During this period, I was constantly trying to not give up on this dissertation's work. After fully grasping the general Blockchain aspects, I then moved to Ethereum and its smart contracts. In the beginning, it was puzzling to be how software could be written without the possibility of further patching. Again, I realized I needed to dedicate 200% if I was to have a slim chance of completing this work.

The GDPR was the next objective in my work. Reading it, understanding it and learning how it can be applied proved to be one of my most difficult challenges I had to face to date, as I am not a lawyer and I needed to start having a mindset of one. Then, there were the many different GDPR articles which I was required to know if I was to tackle the problem of immutability in Ethereum and its smart contracts.

At this point, I understood the different pieces of the puzzle: the GDPR and the Ethereum Blockchain with its smart contracts. I then began to apply my knowledge of standard, centralized software development to Ethereum and its smart contracts: Blockchain Software Engineering. During this period, the DFiles DApp was built.
I also had a substantial amount of issues with DFiles, as I had to develop it from scratch

[6] https://oauth.io/

with almost the entirety of my development stack unknown to me.

As a personal conclusion, I can say I dedicated myself to the maximum extent and enjoyed it every minute. The end result was that I was hired long before I finished this work by a well-established organization with a very important project where I will develop Blockchain software applications and apply all my knowledge acquired throughout this amazing experience.

Bibliographic References

Bauerle, Nolan (Mar. 2017). *What is a Distributed Ledger?* url: https://www.coindesk.com/information/what-is-a-distributed-ledger/.

Bitcoin Project (2018). *Bitcoin - Open Source P2P Money*. url: https://bitcoin.org/en/.

Bitcoin Wiki (Mar. 2018a). *Irreversible Transactions*. url: https://en.bitcoin.it/wiki/Irreversible_Transactions.

– (2018b). *Proof of Work*. url: https://en.bitcoin.it/wiki/Proof_of_work.

BlockchainHub (2018). *Blockchains & Distributed Ledger Technologies*. url: https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/.

Buterin, Vitalik (Jan. 2016). *Privacy on the Blockchain*. url: https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/.

– (Oct. 2017). *Serpent*. url: https://github.com/ethereum/serpent.

Coindesk (Aug. 2015). *What is bitcoin?* url: https://www.coindesk.com/information/what-is-bitcoin/.

Consensys (Jan. 2018). *Truffle Suite*. url: https://github.com/trufflesuite/truffle.

Dannen, Chris (2017). *Introducing Ethereum and Solidity*. 1st ed. Apress, p. 185. isbn: 978-1-4842-2534-9. doi: 10.1007/978-1-4842-2535-6. url: http://link.springer.com/10.1007/978-1-4842-2535-6.

Destefanis, Giuseppe et al. (Mar. 2018). *Smart Contracts Vulnerabilities: A Call for Blockchain Software Engineering?*

dmpldr (2017). *Embark VS Truffle r/ethdev*. url: https://www.reddit.com/r/ethdev/comments/6p8qx6/embark_vs_truffle/.

Encyclopedia Britannica (Jan. 2018). *Supply and Demand*. url: https://www.britannica.com/topic/supply-and-demand.

Ethereum Foundation (2018a). *Account Types, Gas, and Transactions*. url: http://www.ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html.

– (2018b). *Ether*. url: http://www.ethdocs.org/en/latest/ether.html.

– (2018c). *Ethereum Project*. url: https://www.ethereum.org/.

– (2018d). *Ethereum Wiki*. url: https://github.com/ethereum/wiki/wiki/Dapp-using-Meteor.

– (2018e). *Ethereum Wiki*. url: https://github.com/ethereum/wiki/wiki/JSON-RPC.

– (2018f). *Solidity*. url: https://solidity.readthedocs.io/en/develop/.

– (2018g). *Solidity, the Contract-Oriented Programming Language*. url: https://github.com/ethereum/solidity.

– (Feb. 2018h). *Vyper*. url: https://github.com/ethereum/vyper.

– (2018i). *What is Ethereum?* url: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html.

EU GDPR (2018). *Key Changes with the General Data Protection Regulation*. url: https://www.eugdpr.org/key-changes.html.

EU GDPR Compliant (2018). *What is a Data Subject?* url: https://eugdprcompliant.com/what-is-data-subject/.

European Union (May 2016). "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* L119, pp. 1–88. url: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN.

Exodus (2018). *What is an ERC20 Token*. url: https://support.exodus.io/article/108-what-is-an-erc20-token-and-does-exodus-support-it.

Falkon, Samuel (Dec. 2017). *The Story of the DAO — Its History and Consequence*. url: https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee.

ICO (2011). *The Privacy and Electronic Communications (EC Directive) (Amendment) Regulations 2011 - Personal Data Security Breach Log*. url: https://ico.org.uk/media/for-organisations/documents/1724/pecr_personal_data_security_breach_notification_form.xls.

– (Aug. 2018). *Guide to the General Data Protection Regulation (GDPR)*. url: https://ico.org.uk/media/for-organisations/guide-to-the-general-data-protection-regulation-gdpr-1-0.pdf.

Investopedia Staff (Apr. 2016). *Game Theory*. url: https://www.investopedia.com/terms/g/gametheory.asp.

J.P. Morgan (2016). "Quorum". In: pp. 1–8. url: https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf.

Khatwani, Sudhir (Nov. 2017). *NEO vs. Ethereum: How Is Neo Different Than Ethereum?* url: https://coinsutra.com/neo-vs-ethereum-differences/.

– (Jan. 2018). *The Top 10 Best Ethereum Wallets (2018 Edition)*. url: https://coinsutra.com/best-etherum-wallets/.

Koen, Peter (2004). "Understanding the Front End: A Common Language and Structured Picture". In: *Stevens Institute of Technology*.

Koen, Peter et al. (2001). "Providing Clarity and a Common Language To the 'Fuzzy Front End.'" In: *Research Technology Management* 44.2, pp. 46–55. issn: 08956308.

Kosba, Ahmed et al. (2015). "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts". In: pp. 1–31. url: https://eprint.iacr.org/2015/675.pdf.

– (2016). *Hawk: Privacy-Preserving Blockchain and Smart Contracts*. url: http://oblivm.com/hawk/index.html.

Litecoin Project (2018). *Litecoin - Open source P2P Digital Currency*. url: https://litecoin.org/.

Meteor Development Group Inc. (2018). *Introduction*. url: https://guide.meteor.com/.

Miessler, Daniel (May 2011). *Encoding vs. Encryption vs. Hashing vs. Obfuscation*. url: https://danielmiessler.com/study/encoding-encryption-hashing-obfuscation/.

Momoh, Osi (June 2017). *Ripple (Cryptocurrency)*. url: https://www.investopedia.com/terms/r/ripple-cryptocurrency.asp.

NEO Team (2018). *NEO - An Open Network For Smart Economy*. url: https://neo.org/?culture=en-us.

Nicola, Susana, Eduarda Pinto Ferreira, and J. J. Pinto Ferreira (2012). "A Novel Framework for Modeling Value for the Customer, an Essay on Negotiation". In: *International*

Journal of Information Technology & Decision Making 11.03, pp. 661–703. doi: 10.1142/s0219622012500162.

Parity Technologies (July 2017). *The Multi-sig Hack: A Postmortem*. url: https://paritytech.io/the-multi-sig-hack-a-postmortem/.

– (2018). *Parity*. url: https://www.parity.io/.

Petrov, Sergey (Nov. 2017). *Another Parity Wallet Hack Explained*. url: https://medium.com/@Pr0Ger/another-parity-wallet-hack-explained-847ca46a2e1c.

Radziwill, Nicole (Jan. 2016). *Analytic Hierarchy Process (AHP) with the Ahp Package*. url: https://www.r-bloggers.com/analytic-hierarchy-process-ahp-with-the-ahp-package/.

Rosic, Ameer (Dec. 2017a). *Proof of Work vs Proof of Stake: Basic Mining Guide*. url: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/.

– (Dec. 2017b). *What is Bitcoin? A Step-By-Step Guide For Beginners*. url: https://blockgeeks.com/guides/what-is-bitcoin/.

– (Aug. 2017c). *What is Cryptoeconomics? The Ultimate Beginners Guide*. url: https://blockgeeks.com/guides/what-is-cryptoeconomics/.

Rouse, Margaret, Sue Troy, and Mary K. Pratt (Aug. 2017). *What is Distributed Ledger Technology (DLT)?* url: http://searchcio.techtarget.com/definition/distributed-ledger.

Saaty, Thomas L. (2008). "Decision Making with the Analytic Hierarchy Process". In: *Int. J. Services Sciences* 1.1, pp. 83–98. doi: 10.18411/a-2017-023. url: http://www.rafikulislam.com/uploads/resourses/197245512559a37aadea6d.pdf.

Status Research & Development GmbH (Feb. 2018). *Embark Framework*. url: https://github.com/iurimatias/embark-framework.

Szabo, Nick (1996). *Smart Contracts: Building Blocks for Digital Markets*. url: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

The Economic Times (2018). *What is Cryptography?* url: https://economictimes.indiatimes.com/definition/cryptography.

Wöhrer, Maximilian and Uwe Zdun (2018). "Design Patterns for Smart Contracts in the Ethereum Ecosystem". In: p. 8.

Woodall, Tony (Jan. 2003). "Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis". In: 12.

Zeithaml, Valarie (July 1988). "Consumer Perceptions of Price, Quality, and Value: A Means-end Model and Synthesis of Evidence". In: *Journal of Marketing*. 22nd ser. 52.2, pp. 1–21. doi: 10.3897/bdj.4.e7720.figure2f. url: https://hec.unil.ch/docs/files/123/997/zeithaml88-1.pdf.

Appendices

Appendix A

AHP R Code

```r
# The following code is an exact copy of the code available at:\\
https://github.com/gluc/ahp

devtools::install_github("gluc/ahp", build_vignettes = TRUE)
vignette("car-example", package = "ahp")
vignette("multiple-decisionmakers", package = "ahp")

# run analysis
library(ahp)
ahpFile <- system.file("extdata", "car.ahp", package = "ahp")
carAhp <- Load(ahpFile)
Calculate(carAhp)
Visualize(carAhp)
Analyze(carAhp)
AnalyzeTable(carAhp)


# looking at the vacation example, a multi-decision-maker model
ahpFile <- system.file("extdata", "vacation.ahp", package = "ahp")
vacationAhp <- Load(ahpFile)
Calculate(vacationAhp)
Analyze(vacationAhp, decisionMaker = "Dad")
AnalyzeTable(vacationAhp, decisionMaker = "Mom")
AnalyzeTable(vacationAhp)
RunGUI()
```

Listing A.1: The AHP R code.

Appendix B

AHP YAML Code

```yaml
Version: 2.0

#########################
# Alternatives Section
#

Alternatives: &alternatives
# Here, we list all the alternatives, together with their attributes.
# We can use these attributes later in the file when defining
# preferenceFunctions. The attributes can be quantitative or
# qualitative.

# Note: when comparing Ethereum Wallets, tre attributes of each wallet can be ignored.
# For this reason, here the alternatives are specified without any attributes.
  Parity:
  Mist:
  Metamask:

#
# End of Alternatives Section
#####################################

#####################################
# Goal Section
#


Goal:
# The goal spans a tree of criteria and the alternatives
  name: Choose The Best Ethereum Wallet
  description: >
```

```
32      This is a simple decision making program with AHP.
    The objective is
33      to choose the best Ethereum Wallet based on certain
    criteria.
34    author: Duarte Teles
35    preferences:
36      # preferences are typically defined pairwise
37      # 1 means: A is equal to B
38      # 9 means: A is highly preferrable to B
39      # 1/9 means: B is highly preferrable to A
40      pairwise:
41
42      # here, each criterion is compared. FOr example, in
    an Ethereum Wallet,
43      # Performance is highly preferred compared to, for
    instance, features.
44      # The same principle can be applied for others, such
     as privacy VS storage,
45      # for example
46        - [Features, Performance, 1/7]
47        - [Features, Privacy, 1/7]
48        - [Features, Storage, 1/7]
49        - [Security, Performance, 2]
50        - [Security, Privacy, 1]
51        - [Security, Storage, 1]
52        - [Performance, Privacy, 1/2]
53        - [Performance, Storage, 1/3]
54        - [Privacy, Storage, 5]
55    children:
56    # here all the criteria are compared, in a pairwise
    manner, for each Ethereum Wallet
57    # For example, regarding features, Metamask is the
    undisputed king compared to all other
58    # Ethereum Wallets
59      Features:
60        preferences:
61          pairwise:
62            - [Parity, Mist, 4]
63            - [Parity, Metamask, 7]
64            - [Mist, Metamask, 9]
65        children: *alternatives
66      Security:
67        preferences:
68          pairwise:
69            - [Parity, Mist, 1/9]
70            - [Parity, Metamask, 1/9]
71            - [Mist, Metamask, 1]
72        children: *alternatives
73      Performance:
```

```
        preferences:
          pairwise:
            - [Parity, Mist, 6]
            - [Parity, Metamask, 1/3]
            - [Mist, Metamask, 1/9]
        children: *alternatives
      Privacy:
        preferences:
          pairwise:
            - [Parity, Mist, 1]
            - [Parity, Metamask, 1]
            - [Mist, Metamask, 1]
        children: *alternatives
      Storage:
        preferences:
          pairwise:
            - [Parity, Mist, 7]
            - [Parity, Metamask, 1/9]
            - [Mist, Metamask, 1/9]
        children: *alternatives


#
# End of Goal Section
#####################################

```

Listing B.1: The AHP YAML code.

Appendix C

State Machines Ethereum Design Pattern Example

```
1
2  pragma solidity ^0.4.17;
3  contract DepositLock {
4      enum Stages {
5          AcceptingDeposits,
6          FreezingDeposits,
7          ReleasingDeposits
8      }
9      Stages public stage = Stages.AcceptingDeposits;
10     uint public creationTime = now;
11     mapping (address => uint) balances;
12
13     modifier atStage(Stages _stage) {
14         require(stage == _stage);
15         _;
16         }
17
18     modifier timedTransitions() {
19         if (stage == Stages.AcceptingDeposits && now >=creationTime + 1
       days)nextStage();
20         if (stage == Stages.FreezingDeposits && now >=creationTime + 8
       days)nextStage();
21         _;
22         }
23
24     function nextStage() internal {stage = Stages(uint(stage) + 1);
25     }
26
27     function deposit() public payable timedTransitionsatStage(Stages.
       AcceptingDeposits) {
28         balances[msg.sender] += msg.value;
29     }
30
31     function withdraw() public timedTransitions atStage(Stages.
       ReleasingDeposits) {
32         uint amount = balances[msg.sender];balances[msg.sender] = 0;msg.
       sender.transfer(amount);
33     }
34 }
```

Listing C.1: A smart contract based on a state machine "to represent a deposit lock, which accepts deposits for a period of one day and releases them after seven days" (Wöhrer and Zdun 2018).

Appendix D

Ownable Ethereum Design Pattern Example

```
1
2  pragma solidity ^0.4.17;
3  contract Ownable {
4      address public owner;
5
6      event LogOwnershipTransferred(address indexedpreviousOwner, address
       indexed newOwner);
7
8      modifier onlyOwner() {
9          require(msg.sender == owner);
10         _;
11     }
12
13     function Ownable() public {
14         owner = msg.sender;
15     }
16
17     function transferOwnership(address newOwner) public onlyOwner {
18         require(newOwner != address(0));LogOwnershipTransferred(owner,
       newOwner);
19         owner = newOwner;
20     }
21 }
```

Listing D.1: A smart contract to track the ownership of a contract (Wöhrer and Zdun 2018).

Appendix E

Access Restriction Ethereum Design Pattern Example

```
1
2  pragma solidity ^0.4.17;
3  import "./Ownership.sol";
4  contract AccessRestriction is Ownable {
5      uint public creationTime = now;
6      modifier onlyBefore(uint _time) {
7          require(now < _time);
8           _;
9           }
10
11     modifier onlyAfter(uint _time) {
12         require(now > _time);
13         _;
14     }
15
16     modifier onlyBy(address account) {
17         require(msg.sender == account);
18          _;
19          }
20
21     modifier condition(bool _condition) {
22         require(_condition);
23          _;
24          }
25
26     modifier minAmount(uint _amount) {
27         require(msg.value >= _amount);
28          _;
29          }
30
31     function f() payable onlyAfter(creationTime + 1 minutes) onlyBy(owner)
        minAmount(2 ether) condition(msg.sender.balance >= 50 ether) {
32         // some code
33     }
34 }
```

Listing E.1: A smart contract demonstrating how to check certain requirements prior to function execution (Wöhrer and Zdun 2018).

Appendix F

Mortal Ethereum Design Pattern Example

```
1
2  pragma solidity ^0.4.17;
3  import "../authorization/Ownership.sol";contract Mortal is Ownable {
4      function destroy() public onlyOwner {
5          selfdestruct(owner);
6      }
7
8      function destroyAndSend(address recipient) public onlyOwner {
9          selfdestruct(recipient);
10         }
11 }
```

Listing F.1: A smart contract that provides its creator with the ability to destroy it (Wöhrer and Zdun 2018).

Appendix G

Satellite Ethereum Design Pattern Example

```
1
2 pragma solidity ^0.4.17;
3 contract Satellite {
4     function calculateVariable() public pure returns (uint){
5         // calculate var return 2*3;
6     }
7 }
```

Listing G.1: "A satellite contract encapsulates certain contract functionalities" (Wöhrer and Zdun 2018).

```
1
2 pragma solidity ^0.4.17;
3 import "../../authorization/Ownership.sol";import "./Satellite.sol";
4 contract Base is Ownable {
5     uint public variable;
6     address satelliteAddress;
7
8     function setVariable() public onlyOwner {
9         Satellite s = Satellite(satelliteAddress);variable = s.
    calculateVariable();
10    }
11
12    function updateSatelliteAddress(address _address) publiconlyOwner {
13        satelliteAddress = _address;
14    }
15 }
```

Listing G.2: A base smart contract "referring to a satellite contract in order to fulfil its purpose. The use of a satellite allows an easy contract functionality modification" (Wöhrer and Zdun 2018).

Appendix H

Other Blockchains and Cryptocurrencies

H.1 Neo

Neo is a non-profit community-based project that "utilizes Blockchain technology and digital identity to digitize assets, to automate the management of digital assets using smart contracts, and to realize a 'smart economy' with a distributed network".
In short, NEO has three main components (NEO Team 2018):

- **Digital Assets:** "Digital Assets are programmable assets that exist in the form of electronic data. With Blockchain technology, the digitization of assets can be decentralized, trustful, traceable, highly transparent, and free of intermediaries. On the NEO Blockchain, users are able to register, trade, and circulate multiple types of assets. Proving the connection between digital and physical assets is possible through digital identity. Assets registered through a validated digital identity are protected by law
- **Digital Identity:** Digital identity refers to the identity information of individuals, organizations, and other entities that exist in electronic form. The more mature digital identity system is based on the Public Key Infrastructure (PKI) X.509 standard. In NEO, we will implement a set of X.509 compatible digital identity standards. This set of digital identity standards, in addition to a compatible X.509 level certificate issuance model, will also support the Web Of Trust point-to-point certificate issuance model
- **Smart Contract:** The NeoContract smart contract system is the biggest feature of the seamless integration of the existing developer ecosystem. Developers do not need to learn a new programming language, but can use C#, Java and other mainstream programming languages in their familiar IDE environments (Visual Studio, Eclipse, etc.) for smart contract development, debugging and compilation. NEO's Universal Lightweight Virtual Machine, NeoVM, has the advantages of high certainty, high concurrency, and high scalability. The NeoContract smart contract system will allow millions of developers around the world to quickly carry out the development of smart contracts"

Differences and Similarities from Ethereum

Recall from chapter 4 the Ethereum Blockchain. Comparative to NEO, it has the following differences[1]:

1. **Smart Contracts, ICOs and DApps:** both Blockchains are designed to host them, in a decentralized way
2. **Similar Cryptoassets:** NEO has GAS, Ethereum Ether
3. **Open Source:** both are open source and Turing complete

According to Khatwani (2017), the differences include:

- **Backers:**
  - "NEO is backed by the Chinese government. This has become the most important factor for NEO's popularity in China, making it China's first open-source public Blockchain project. NEO is also backed by WINGS, Alibaba, and various Microsoft-like giants
  - Ethereum is not backed by any nation's government, but it is supported by the EEA-Enterprise Ethereum Alliance. This speaks volumes about its popularity and potential success on the world's stage
- **Consensus mechanism:**
  - NEO uses a delegated Byzantine Fault Tolerant (dBFT) consensus mechanism, which is an improved form of proof-of-stake
  - Ethereum uses a proof-of-work mechanism
- **Divisibility:**
  - The native crypto-fuel of NEO's Blockchain is not divisible. It only exists in whole numbers (1,2,50,1000, etc.)
  - The native crypto-fuel of Ethereum is divisible. It can be divided (3.4352, 283.399403, etc.)
- **Fueling the Blockchain:**
  - NEO produces a special crypto asset called NeoGAS (or GAS) which is used as a crypto-fuel for NEO's Blockchain
  - In Ethereum, small units of Ether are used as 'gas' to fuel the network
- **Language:**
  - NEO's smart contracts and DApps can be written and compiled in C# and Java. In the future, developers will also be able to write smart contracts in Python and Go. This will drastically reduce the entry barrier for all developers around the world

[1] https://coinsutra.com/neo-vs-ethereum-differences/

  - Currently, if you want to write smart contracts and DApps on Ethereum, then you have to learn a new programming language called Solidity. Very few people know this language

- **Direction:**
  - NEO is focused on making a smart economy by digitizing traditional real world assets via digital identity, along with running smart contracts and DApps
  - Ethereum is going in the direction of becoming the world's only supercomputer based on the Blockchain by hosting numerous use cases of digital identity, computing, decentralized exchanges and remittances
- **Speed:**
  - At present, NEO can handle 10,000 transactions per second
  - Ethereum can handle 15 transactions per second
- **Quantum Computer-proof:**
  - NEO is quantum computer-proof. Quantum computers are believed to have the ability to break into and hack the cryptographic math on which Blockchains are based
  - Ethereum, and every other cryptocurrency, is not quantum computer-proof"

Finally, some additional details about this cryptocurrency, such as **market capitalization**, **total supply** and more in Table 35.

Table 35: Neo data courtesy of **Coin Market Cap**.

Neo		
ID	**Symbol**	**Cryptocurrency Rank**
neo	NEO	15
Price In US$	**Market Capitalization in US$**	
$15.74	$1,023,364,169.00	
Maximum Supply	**Total Supply**	
100,000,000	100,000,000	

H.2 Litecoin

Litecoin is a peer-to-peer currency which enables instant, "near-zero cost payments to anyone in the world. Litecoin is an open source, global payment network that is fully decentralized without any central authorities. Mathematics secures the network and empowers individuals to control their own finances. Litecoin features faster transaction confirmation times and improved storage efficiency than the leading math-based currency" (Litecoin Project 2018).

Litecoin VS Ethereum

Table 36 shows the differences between litecoin and Ether.

Table 36: litecoin VS Ethereum, from **BitDegree**.

	Ether	**Litecoin**
Total Supply of Coins	Unclear	84,000,000
Count Available Supply	98,310,400	55,714,900
Average Transaction Cost	$0.21	$0.21
Average Block Time	10-12 Seconds	2.5 Minutes
Proof-of-Work Algorithm	Ethash	Scrypt
Primary Use	Creating DApps	Payment and test network for Bitcoin
Founder	Vitalik Buterin	Charlie Lee

On top of that, some other interesting details about litecoin — Table 37.

Table 37: Litecoin data courtesy of **Coin Market Cap**.

Litecoin		
ID	**Symbol**	**Cryptocurrency Rank**
litecoin	LTC	7
Price In US$	**Market Capitalization in US$**	
$53.44	$3,137,242,846.00	
Maximum Supply	**Total Supply**	
84,000,000	58,711,127	

H.3 Ripple

Ripple is a technology that acts as both a "cryptocurrency and a digital payment network for financial transactions.
Ripple was released in 2012 and co-founded by Chris Larsen and Jed McCaleb.
The coin for the cryptocurrency is premined and labeled XRP" (Momoh 2017).

Ripple vs Ethereum (ETH)

Table 38 represents the differences between Ethereum (ETH) and ripple.

Table 38: Ripple VS Ethereum, from **UnHashed**.

Cryptocurrency	**Ripple**	**Ethereum (ETH)**
Release Date	2012	2015
Mining Algorithm	None	Ethash (PoW)
Blockchain Programming Language	C++	Go, C++, Rust, Solidity
Avg Blocktime	3.5s	14.2 sec

Last but not least, some interesting details about ripple — Table 39.

Table 39: Ripple data courtesy of **Coin Market Cap**.

Ripple		
ID	**Symbol**	**Cryptocurrency Rank**
ripple	XRP	3
Price In US$	**Market Capitalization in US$**	
$0.42	$16,769,298,594.00	
Maximum Supply	**Total Supply**	
100,000,000,000	99,991,817,275	

Appendix I

Files.sol Complete Smart Contract Source Code

```
1
2  pragma solidity 0.4.24;
3
4  import "./zeppelin/ownership/Ownable.sol";
5
6  /// @title Files
7  /// @author miniclip22
8  /// @notice This smart contract stores all the data extracted from a user
       uploaded file to IPFS. This includes: name, extension, size; IPFS hash
       and a timestamp (when the file was uploaded)
9
10 contract Files is Ownable {
11
12     // contract owner variable
13     address contractOwner;
14
15     // event for when file is added
16     event AddFile(string _name, string _hash, string _extension, uint32
       _size, string _timestamp, address _ethereumAccount);
17
18     // A structure that contains information about the user's files
19     struct File {
20         string name;
21         string extension;
22         uint32 size;
23         string hash;
24         string timestamp; // the transaction timestamp is stored in the
       JavaScript and is in the Unix Epoch time;
25         address ethereumAccount;
26       }
27
28
29     //The files belonging to each user (or address)
30     mapping(address => File[]) private userFiles;
31
32     /// @notice Contract constructor. Here, the contract owner variable is
        assigned the msg.sender value.
33     constructor() public
34       {
35         contractOwner = msg.sender;
36       }
37
```

```
38      /// @notice This function inserts information of a file to the
        userFiles array: name, size, extension in addition to a timestamp and
        the IPFS file hash
39      /// @param _name file name
40      /// @param _hash IPFS file hash
41      /// @param _extension file extension
42      /// @param _size file size
43      /// @param _timestamp a timestamp
44      /// @param _ethereumAccount the user Ethereum account calling this
        contract
45      function addFile(string _name, string _hash, string _extension, uint32
         _size, string _timestamp, address _ethereumAccount)
46      public {
47
48          require(_size>0 && _size<=4294967295, "Invalid file size");
49          File memory file = File(_name, _extension,  _size, _hash,
        _timestamp, _ethereumAccount);
50          userFiles[contractOwner].push(file);
51
52          // emit AddFile event
53          emit AddFile(_name, _hash, _extension, _size, _timestamp,
        _ethereumAccount);
54
55        }
56
57      /// @notice This function gets all properties from the userFiles array
58      /// @param index loop array index
59      /// @return The properties of a file: name, extension, size, IPFS file
         hash and a timestamp
60      function getFileAtIndex(uint256 index) public view returns(string,
        string, uint32, string,  string, address)
61      {
62          File storage file = userFiles[contractOwner][index];
63          return (file.name, file.extension, file.size, file.hash, file.
        timestamp, file.ethereumAccount);
64        }
65
66      /// @notice This function gets the length of the userFiles array, so
        it can be looped in the frontend. This way, we avoid having costly
        loops in Solidity code
67      /// @return The length of the userFiles array
68      function getUserFilesLength() public view returns (uint256)
69      {
70          return userFiles[contractOwner].length;
71        }
72    }
```

Listing I.1: Files.sol complete smart contract source code.

Appendix J

Files.sol Smart Contract Unit Tests File

```
const Files = artifacts.require("Files");
const assert = require("assert");
let {
  sha3_512
} = require("js-sha3");

const filesNotFromBlockchain = [{
  name: "Master's Degree Sample",
  extension: ".pdf",
  size: 345435435,
  hash: sha3_512("Master's Degree Sample hash"),
  timestamp: (+new Date()).toString(),
  ethereumAddress: "0x0000000000000000000000000000000000000000",
  // metadata: "",
}];

contract("Files", function (accounts) {
  let filesContractInstance;
  beforeEach(async () => {
    filesContractInstance = await Files.deployed();
  });

  console.log("accounts: ", accounts);

  it("should be possible to store a file's properties in the Blockchain!"
    // And it should have a length of 1"
    , async () => {
      await filesContractInstance.addFile(
        filesNotFromBlockchain[0].name,
        filesNotFromBlockchain[0].hash,
        filesNotFromBlockchain[0].extension,
        filesNotFromBlockchain[0].size,
        filesNotFromBlockchain[0].timestamp,
        filesNotFromBlockchain[0].ethereumAddress, {
          from: accounts[0]
        }
      );

      let fileRetrievedFromSmartContract = await filesContractInstance.
    getFileAtIndex(
        0
      );
```

```
42
43        const fileFromBlockchain = {
44          name: fileRetrievedFromSmartContract[0],
45          extension: fileRetrievedFromSmartContract[1],
46          size: fileRetrievedFromSmartContract[2].toNumber(),
47          hash: fileRetrievedFromSmartContract[3],
48          timestamp: fileRetrievedFromSmartContract[4],
49          ethereumAddress: fileRetrievedFromSmartContract[5],
50        };
51
52        assert.equal(
53          JSON.stringify(filesNotFromBlockchain[0]),
54          JSON.stringify(fileFromBlockchain),
55          "File was not added correctly to the Blockchain :("
56        );
57
58        const filesNotFromBlockchainLength = filesNotFromBlockchain.length.
      toString();
59        const fileFromBlockchainLength = await filesContractInstance.
      getUserFilesLength();
60        assert.equal(
61          filesNotFromBlockchainLength,
62          fileFromBlockchainLength.toString(),
63          "Something is wrong, there should be ONLY one added file in the
      Blockchain :("
64        );
65      });
66
67    it("should NOT be possible to retrieve nonexistent files from the
      Blockchain!!!", async () => {
68      const invalidFileIndex = 99;
69
70      await filesContractInstance.addFile(
71        filesNotFromBlockchain[0].name,
72        filesNotFromBlockchain[0].hash,
73        filesNotFromBlockchain[0].extension,
74        filesNotFromBlockchain[0].size,
75        filesNotFromBlockchain[0].timestamp,
76        filesNotFromBlockchain[0].ethereumAddress, {
77          from: accounts[0]
78        }
79      );
80
81      try {
82        await filesContractInstance.getFileAtIndex(invalidFileIndex);
83        assert.ok(true, `file  is valid. `);
84      } catch (error) {
85
86        assert.equal(
87          error,
88          `Error: VM Exception while processing transaction: invalid opcode`
89        );
90      }
91    });
92  });
```

Listing J.1: Files Ethereum smart contract unit testing file.

Appendix K

DFiles Encrypted Files Statistical Data

Table 40: DFiles collected data for statistical analysis: encrypted files.

Small Files (1KB-1MB)					
Gas Fee (ETH)	**Gas Fee (USD)**	**Amount + Gas Fee (Total) (ETH)**	**Amount + Gas Fee (Total) (USD)**	**Size (MB)**	**Uploaded Elapsed Time (Seconds)**
0.000426	0.09	0.000426	0.09	0.013	5
0.000426	0.09	0.000426	0.09	0.0136	3
0.000428	0.09	0.000428	0.09	0.0137	3
0.000364	0.08	0.000364	0.08	0.0155	4
0.000428	0.09	0.000428	0.09	0.0162	4
0.000364	0.08	0.000364	0.08	0.0192	4
0.000364	0.08	0.000364	0.08	0.01005	4
0.000364	0.08	0.000364	0.08	0.02601	5
0.000427	0.09	0.000427	0.09	0.027	4
0.000365	0.08	0.000365	0.08	0.03004	4
0.000427	0.09	0.000427	0.09	0.04001	4
0.000365	0.08	0.000365	0.08	0.0591	4
0.000365	0.08	0.000365	0.08	0.06	3
0.000365	0.08	0.000365	0.08	0.067	4
0.00046	0.1	0.00046	0.1	0.0827	4
0.000426	0.09	0.000426	0.09	0.0842	5
0.000365	0.08	0.000365	0.08	0.1265	4
0.000364	0.08	0.000364	0.08	0.1848	4
0.000366	0.08	0.000366	0.08	0.193	5
0.000365	0.08	0.000365	0.08	0.2491	5
0.000366	0.08	0.000366	0.08	0.6608	5
0.000429	0.09	0.000429	0.09	0.7325	5
0.000365	0.08	0.000365	0.08	0.7349	5
0.000364	0.08	0.000364	0.08	0.9445	5
0.000364	0.08	0.000364	0.08	0.984	5
Medium Files (1MB-20MB)					
0.000365	0.08	0.000365	0.08	1.2	6
0.000427	0.09	0.000427	0.09	1.3	7
0.000426	0.09	0.000426	0.09	1.5	7
0.000365	0.08	0.000365	0.08	1.7	8
0.000426	0.09	0.000426	0.09	2.2	9
0.000428	0.09	0.000428	0.09	2.3	11
0.000364	0.08	0.000364	0.08	3.9	16
0.000364	0.08	0.000364	0.08	4.5	17
0.000426	0.09	0.000426	0.09	4.8	21
0.000363	0.08	0.000363	0.08	4.8	14
0.000427	0.09	0.000427	0.09	6.7	22
0.000427	0.09	0.000427	0.09	8	26
0.000365	0.08	0.000365	0.08	10.3	50
0.000428	0.09	0.000428	0.09	11.7	49
0.000364	0.08	0.000364	0.08	12	69
0.000364	0.08	0.000364	0.08	12.2	65
0.000364	0.08	0.000364	0.08	13.9	69
0.000396	0.08	0.000396	0.08	14.2	66

Appendix L

DFiles Unencrypted Files Statistical Data

Table 41: DFiles collected data for statistical analysis: unencrypted files.

Small Files (1KB-1MB)					
Gas Fee (ETH)	**Gas Fee (USD)**	**Amount + Gas Fee (Total) (ETH)**	**Amount + Gas Fee (Total) (USD)**	**Size (MB)**	**Uploaded Elapsed Time (Seconds)**
0.00044	0.1	0.00044	0.1	0.013	3
0.000438	0.1	0.000438	0.1	0.0136	3
0.00044	0.1	0.00044	0.1	0.0137	2
0.000375	0.08	0.000375	0.08	0.0155	2
0.000439	0.1	0.000439	0.1	0.0162	2
0.000375	0.08	0.000375	0.08	0.0192	3
0.000375	0.08	0.000375	0.08	0.01005	2
0.000375	0.08	0.000375	0.08	0.02601	4
0.000438	0.1	0.000438	0.1	0.027	3
0.000377	0.08	0.000377	0.08	0.03004	4
0.000438	0.1	0.000438	0.1	0.04001	3
0.000376	0.08	0.000376	0.08	0.0591	4
0.000376	0.08	0.000376	0.08	0.06	3
0.000472	0.11	0.000472	0.11	0.067	4
0.000438	0.1	0.000438	0.1	0.0827	4
0.000376	0.08	0.000376	0.08	0.0842	4
0.000375	0.08	0.000375	0.08	0.1265	5
0.000377	0.08	0.000377	0.08	0.1848	4
0.000376	0.08	0.000376	0.08	0.193	4
0.000377	0.08	0.000377	0.08	0.2491	4
0.000441	0.1	0.000441	0.1	0.6608	4
0.000376	0.08	0.000376	0.08	0.7325	4
0.000375	0.08	0.000375	0.08	0.7349	4
0.000376	0.08	0.000376	0.08	0.9445	5
0.000439	0.1	0.000439	0.1	0.984	5
Medium Files (1 MB - 20MB)					
Continued on next page					

Table 41 – continued from previous page

Gas Fee (ETH)	Gas Fee (USD)	Amount + Gas Fee (Total) (ETH)	Amount + Gas Fee (Total) (USD)	Size (MB)	Uploaded Elapsed Time (Seconds)
0.000376	0.08	0.000376	0.08	1.2	5
0.000438	0.1	0.000438	0.1	1.3	5
0.000438	0.1	0.000438	0.1	1.5	5
0.000376	0.08	0.000376	0.08	1.7	5
0.000438	0.1	0.000438	0.1	2.2	5
0.000438	0.1	0.000438	0.1	2.3	6
0.00044	0.1	0.00044	0.1	3.9	6
0.000375	0.08	0.000375	0.08	4.5	6
0.000438	0.1	0.000438	0.1	4.8	6
0.000375	0.08	0.000375	0.08	4.8	6
0.000438	0.1	0.000438	0.1	6.7	6
0.000439	0.1	0.000439	0.1	8	9
0.000376	0.08	0.000376	0.08	10.3	7
0.000439	0.1	0.000439	0.1	11.7	7
0.000375	0.08	0.000375	0.08	12	6
0.000375	0.08	0.000375	0.08	12.2	8
0.000375	0.08	0.000375	0.08	13.9	8
0.000407	0.09	0.000407	0.09	14.2	7
0.000439	0.1	0.000439	0.1	15.2	8
0.000375	0.08	0.000375	0.08	16.1	6
0.000441	0.1	0.000441	0.1	16.4	8
0.000376	0.08	0.000376	0.08	18	9
0.00044	0.1	0.00044	0.1	18.8	8
0.000375	0.08	0.000375	0.08	18.9	6
0.000439	0.1	0.000439	0.1	18.9	10
Large Files (20MB - 200MB)					
0.000439	0.1	0.000439	0.1	20.8	12
0.000438	0.1	0.000438	0.1	21.1	9
0.000375	0.08	0.000375	0.08	23.2	9
0.000407	0.09	0.000407	0.09	24.6	7
0.00044	0.1	0.00044	0.00044	26.5	7
0.00044	0.1	0.00044	0.1	26.8	13
0.000438	0.1	0.000438	0.1	29.6	11
0.000376	0.09	0.000376	0.09	33.8	10
0.000375	0.08	0.000375	0.08	37.6	11
0.000377	0.09	0.000377	0.09	37.8	9
0.000375	0.08	0.000375	0.08	44	13
0.000375	0.08	0.000375	0.08	50.2	13
0.000472	0.11	0.000472	0.11	52.9	17
0.000472	0.11	0.000472	0.11	55.8	19
0.000472	0.11	0.000472	0.11	66.8	18

Continued on next page

Table 41 – continued from previous page

Gas Fee (ETH)	Gas Fee (USD)	Amount + Gas Fee (Total) (ETH)	Amount + Gas Fee (Total) (USD)	Size (MB)	Uploaded Elapsed Time (Seconds)
0.000375	0.08	0.000375	0.08	67.5	17
0.000375	0.08	0.000375	0.08	74.9	13
0.000439	0.1	0.000439	0.1	75.4	20
0.000473	0.11	0.000473	0.11	87.7	12
0.000439	0.1	0.000439	0.1	101.9	14
0.000375	0.08	0.000375	0.08	129.9	5
0.000375	0.08	0.000375	0.08	167.9	5
0.000375	0.08	0.000375	0.08	178.8	5
0.000375	0.08	0.000375	0.08	179.2	8
0.000375	0.08	0.000375	0.08	185.7	9
Extra Large Files (200 MB – 2 GB)					
0.000375	0.08	0.000375	0.08	231.1	9
0.000375	0.08	0.000375	0.08	321.6	6
0.000375	0.08	0.000375	0.08	339	10
0.000375	0.08	0.000375	0.08	395.7	11
0.000375	0.08	0.000375	0.08	448.3	11
0.000375	0.08	0.000375	0.08	475	13
0.000375	0.08	0.000375	0.08	504.8	15
0.000375	0.08	0.000375	0.08	528	24
0.00042	0.09	0.00042	0.09	548.9	25
0.00042	0.09	0.00042	0.09	657.2	28
0.00042	0.09	0.00042	0.09	732	54
0.00042	0.09	0.00042	0.09	898.4	107

Appendix M

DFiles descriptive analysis and comparison of small files

Table 42: Descriptive analysis and comparison small files.

Amount + Gas fee (total) (ETH)		
Descriptive measures	**Unencrypted**	**Encrypted**
Minimum	0.000375	0.000364
Maximum	0.000472	0.00046
Mean	0.0004024	0.0003885
1st quartile	0.000376	0.000364
Median	0.000377	0.000365
3rd quartile	0.000438	0.000426
Standard deviation	0.0000339	0.000033
Coefficient of variation	8.40%	8.50%
Skewness	0.51	0.68
Shapiro-Wilk test		
Test statistic	0.7	0.68
p-value	0.001	0.001
Wilcoxon test		
Test statistic	262	
p-value	0.007	
Uploaded elapsed time (seconds)		
Descriptive measures	**Unencrypted**	**Encrypted**
Minimum	2	3
Maximum	5	5
Mean	3.6	4.3
1st quartile	3	4
Median	4	4
3rd quartile	4	5
Standard deviation	0.92	0.68
Coefficient of variation	25.70%	15.80%
Skewness	-0.33	-0.36
Shapiro-Wilk test		
Test statistic	0.86	0.78
p-value	0.003	0.0001
Wilcoxon test		
Test statistic	6.5	
p-value	0.001	
Size (MB)		
Descriptive measures	**Encrypted and Unencrypted**	
Minimum	0.01005	
Maximum	0.984	
Mean	0.2155	
1st quartile	0.0192	
Median	0.06	
3rd quartile	0.193	
Standard deviation	0.31577	
Coefficient of variation	146.50%	
Skewness	1.4	

Appendix N

DFiles descriptive analysis and comparison of medium files

Table 43: Descriptive analysis and comparison medium files.

Amount + Gas fee (total) (ETH)		
Descriptive measures	**Unencrypted**	**Encrypted**
Minimum	0.0003750	0.0003640
Maximum	0.0004400	0.0004280
Mean	0.0004087	0.0003938
1st quartile	0.0003753	0.0003640
Median	0.0004225	0.0003805
3rd quartile	0.0004380	0.0004268
Standard deviation	0.0000315	0.0000313
Coefficient of variation	7.7%	7.9%
Skewness	-0.10	0.10
Shapiro-Wilk test		
Test statistic	0.69	0.69
p-value	< 0.001	< 0.001
Wilcoxon test		
Test statistic	171	
p-value	0.0002	
Uploaded elapsed time (seconds)		
Descriptive measures	**Unencrypted**	**Encrypted**
Minimum	5	6
Maximum	9	69
Mean	6.3	29.6
1st quartile	6	10
Median	6	19
3rd quartile	7	50
Standard deviation	1.2	24.3
Coefficient of variation	18.8%	82.2%
Skewness	0.71	0.63
Shapiro-Wilk test		
Test statistic	0.87	0.81
p-value	0.019	0.002
Wilcoxon test		
Test statistic	0	
p-value	0.0002	
Size (MB)		
Descriptive measures	**Encrypted and Unencrypted**	
Minimum	1.2	
Maximum	14.2	
Mean	6.5	
1st quartile	3	
Median	5	
3rd quartile	12	
Standard deviation	4.7	
Coefficient of variation	72.3%	
Skewness	0.37	

Appendix O

DFiles Privacy Policy

The Privacy Policy reflects all the GDPR's key obligations and rights. As a reminder, the privacy policy must be intelligible, easy to understand, concise and written in clear and plain language.
In this chapter, the DFiles Privacy Policy is drafted and presented (adapted from the **the aa** Privacy Policy[1]).
Note that the language used is too informal, simple and direct, as required by the GDPR.

Background

This privacy notice let's you understand what happens to any personal data that is given to us, DFiles Inc, or any that we may collect from or about you. It applies exclusively to the DFiles DApp.
This privacy notice applies to personal information processed by or on behalf of DFiles Inc.

The DFiles Inc and our representative

We are DFiles Inc, a fictional company located in the United Kingdom. We are a data controller and we also process your personal data.
We do not have a Data Protection Officer, due to our small size. Instead, we have Mr. John Smith, our representative, who is responsible for protecting your data.

What kinds of personal information about you do we process?

Personal information that we will process in connection with the DFiles DApp, if relevant, includes:

- Your **email** address
- Your **Ethereum Address**
- Your unique **username**
- Your hashed **password**

[1] http://www.theaa.com/privacy-notice

- Your unique and auto-generated RSA private/public key pair, which is used to encrypt and decrypt your uploaded files to an IPFS node

What is the source of your personal information?

We will collect personal information from the following general sources:

- From you directly
- Information generated about you when you use our DApp

What do we use your personal data for?

We use your personal data, including any of the personal data listed above, for the following purposes:

- User login verification
- User registration
- Encrypting your uploaded files
- Decrypting and viewing your files
- Updating your information
- To perform and/or test the performance of DFiles and internal processes
- To follow guidance and best practice under the change to rules of governmental and regulatory bodies
- To comply with legal and regulatory obligations, requirements and guidance

What are the legal grounds for our processing of your personal information?

- To comply with our legal obligations
- With your consent or explicit consent:
  - For registering your account
  - For encrypting your files
  - For decrypting your files

When do we share your personal information with other organizations?

We currently do not share your information.

How and when can you withdraw your consent?

Currently, you cannot. In the future, we will be implementing new processes to allow consent withdrawal by contacting us.

Is your personal information transferred outside the UK?

No.

What should you do if your personal information changes?

You should update your personal information in the DFiles account settings page.

Do you have to provide your personal information to us?

We cannot provide you with our DFiles features if you do not provide certain information to us. In cases where providing some personal information is optional, it will be clear.

For how long is your personal information retained by us?

Unless we explain otherwise to you, we will hold your personal information based on the following criteria:

- For as long as you upload files using our platform and until you delete your account
- Retention periods in line with legal and regulatory requirements or guidance

What are your rights under data protection laws?

Here is a list of the rights that all individuals have under data protection laws. They do not apply in all circumstances:

- The right **to be informed** about the processing of your personal information
- The right to have your personal information **corrected if it is inaccurate** and to have **personal information information completed**
- The right to **have your personal information erased** (the "right to be forgotten")
- The right to **request access** to your personal information and to obtain information about how we process it
- The right to **move, copy or transfer your personal information** ("data portability")