Technik
Informatik & Medien
Hochschule Ulm
University of
Applied Sciences

# Decentralized Document Management

Master's thesis at the
University of Applied Sciences Ulm
Department of Computer Science
Degree Course Information Systems

submitted by
**Emmanuel SCHWARTZ**
Mat.-Nr: 3119342

April 2017

| | |
|---|---|
| 1st Evaluator: | Prof. Dr. rer. nat. Stefan Traub |
| 2nd Evaluator: | Prof. Dr. rer. nat. Markus Schäffter |

# Declaration of Originality

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

3$^{\text{rd}}$ April, 2017 Emmanuel SCHWARTZ

# Abstract

# Acknowledgements

I would like to express the deepest apprecetion to my first evaluator, Prof. Dr. rer. nat. Stefan Traub, who has always his office open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but didn't hesitate to put me in the right the direction whenever he thought I needed it.

I would also to give a special thanks to Prof. Dr. rer. nat. Markus Schäffter for his spontaneous agreement to act as the second evaluator. I am gratefully indebted to him for his very valuable comments on this thesis.

I owe many thanks also to my colleague and now friend, Florian Schneider for his collaboration. Some parts of our respective thesis required a lot of team work. I'm so grateful that we could achieve a fantastic job together.

Last but not least, I must express my very profound gratitude to my parents and to my girlfriend Jill, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you!

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

It is often considered that the history of electronic mail (or e-mail) begins in 1965, at a time when the Internet did not even exist yet. By that time, the first exchanges of messages was only possible between users on private networks were set up. One of the first systems to allow message exchange was the Competent Time-Sharing System (CTSS) of the famous Massachusetts Institute of Technology (MIT), although this paternity has also been claimed by System Development Corporation SDC) and its own Time-Sharing System created for the Q32, a computer specially manufactured by IBM for the US Air Force.

However, e-mail is only really born from the creation of the ARPAnet network, the ancestor of the Internet. In 1971, after writing some 200 lines of code in order to create two applications, SNDMSG nas READMAIL, the engineer named Raymond Samuel Tomlinson could sent the first email of history to himself. Some times later, Tomlinson found a way for the program to easily differentiate a local message from a network message: the symbol @ was born. It was a simple way to dissociate a user name and host name with the only character that was not used in any proper name nor, above all, in any company name. The first "netmail" test was sent with only content "QWERTYUIOP", the first line of character of the English keyboard.

The email was so successful that it quickly became unthinkable for users of the ARPAnet network to do without it. As a result, the software quickly became the "killer app" of the ARPAnet network, and developers focused either on improving the program and its transfer protocol, or creating their own solutions. In 1992, a great improvment was made: the world's first-ever email attachment, sent by the researcher Nathaniel Borenstein, where we could see a adorable photo of his barbershop quartet, The Telephone Chords. This was made possible thanks to MIME (Multipurpose Internet Mail Extensions),a internet standard that extends the data format of e-mails.

Fourty years later, despite the creation of Instant Messaging, or some years later, social networks, e-mails are still very popular: 183 billion of them are sent every day! If e-mails spam remains a major problem, e-mails has to face new challenges: **Reliability** & **Privacy**.

When a person is sending an e-mail, she expects that her message will be received successfully by the intended recipient. For most cases, it does, but sometimes, for the following reasons, it does not:

–The design of e-mail: Two users does not have to be online at the same time in order to communicate. This is called asynchronious communications. This is made possible by the mail servers, accepting messages from sources and attempt to relay them towards, or deliver them to, the recipient. In order to do so, e-mails have to jump from an server to an other: Some e-mails might get lost during these operations, for various reasons (busy servers, e-mails deferral, rejected e-mails)

–The exponential groth of e-mail spam has forced the use of e-mail rejection, intended to identify and separate legitimate e-mails from junk e-mails. Unfortunately, this has turned out into a false positive problem: Some legimate e-mails are considered as junk, this means the user thinks he did not receive the e-mail. Some solutions try to solve these problems such as RE: Reliable Email[*], which tries to create an intelligent filter for e-mails, or tools that responds automatically to undeliverability by persisting with retransmission or retransmitting to alternate recipients [*]

Alongside with **reliability**, e-mail is facing one of the biggest challenges: **Privacy**. Since the shattering revelations of Edward Snowden, brigging to light the massive world surveillance by the U.S, people try to protect themselves by encypting their communications when possible. The famous free email application, Mozilla Thunderbird has an extension called EnigMail, which can encrypt e-mails thanks to OpenPGP. The downside is, that every user needs to have this add-ons installed, otherwise it will be not possible to read the e-mail that was intented for him. A solution could be that Thunderbird integrates directly this feature in his client, but is still not planned on the road map. But a recent piece of news[*] unviels that the IT-idustry giants such as Google, Microsoft, Yahoo!, LinkedIn and Comcast are working together to elaborate a new encrypted messaging protocol, named SMTP STS (Strict Transport Security)[*]. The idea is that a session starts in clear, and after the server announces that it supports the encrypted connection, the client can then switch to encryption mode, to avoid man-in-the-middle attack. Unfornately no released date was announced yet.

Last downside of e-mails involves the user. Nowadays, it is common practice to send business documents via e-mail or via supplier's web portal: Faster and cheaper compared to a hard copy sent via post. But, how can a user successuffly order and sort out differents kinds of documents, coming from different mailboxes? In most cases, the user does not have a clue how to establish a **document management**: local storage? cloud storage? And what if the computer has hard drive failures, or is infected by viruses: Did they make regular back-ups?

With respect to the application senario described in the next section, this thesis will try to answer these problems with new upcoming technologies such as blockchains, decentralized storage and Dapps.

## 1.2 Application Scenario

Even if the big actors of Internet are trying to adapt e-mails to nowadays requirements, e-mails might not be the right solution anymore to send buisiness documents. Figure 1.1

shows the path that an e-mail takes to go from the company to a client, jumping through servers via the protocol SMTP. Once the email has eached the client's mail server, the client has the choice to synchronize his email between 3 protocols: IMAP, POP3 or SMTP. New technologies that Bitcoin has brought, paired up with decentralized storage providers, can revolutionize the way we are going to send documents.

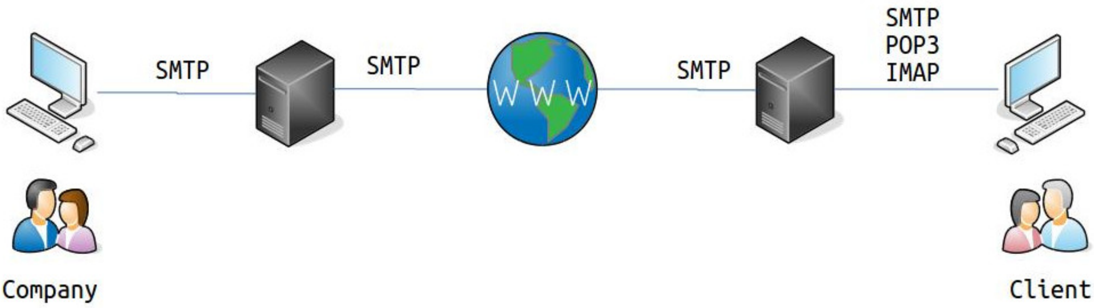

Figure 1.1: Typical path of an e-mail from company to client

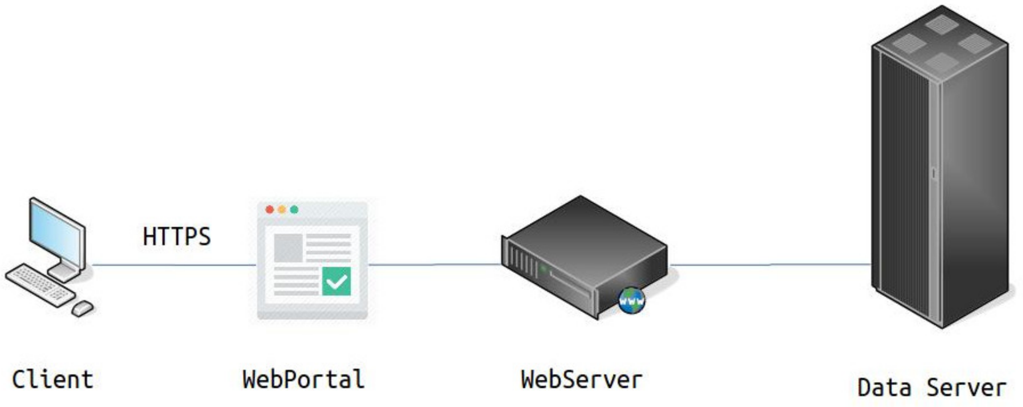

Figure 1.2: Companies WebPortal

dzadazdazdzaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa alsa nskl anslk ansl asn kalsn alsn akls anls anksa nslk ansk alsna lsn aklsn aklsn akl anslka nska lns lasn ksn kls nska lnsaklsnaklsnakslanskla nskla nka naklsnklsnkalnksal sna klsna kslanska lnsakls anskla ns aklsna klsna skl ansa kls ankslnsa kn ans ksnak sna klsna sn a nsa klsnklsnalsa nsla nskla sn aklsna klsnkls nsa ksla nsk lsn lk sanl asn as klsna lks as na slka nskl ans lk

## 1.3 Objective

What do we want to achieve?

## 1.4 Overview

Presentation of the upcoming chapters.

# Chapter 2

# Related Work and Basic Principles

## 2.1 Related Work

### 2.1.1 Towards Cloud-Based Decentralized Storage for Internet of Things Data

### 2.1.2 Prototype of cloud based document management for scientific work validation

## 2.2 Basic Principles

### 2.2.1 Blockchains

In 2008, Bitcoin, the famous cryptocurrency was created, it brought at the same time a new concept: The system operates without central authority or single administrator, but in a decentralized way thanks to the consensus of all the nodes of the network. Based on this Idea, Ethere

#### 2.2.1.1 What is a blockchain?

#### 2.2.1.2 Cryptography, Hash and Signature

#### 2.2.1.3 Transactions

#### 2.2.1.4 Proof of Work

#### 2.2.1.5 Merkle Tree

#### 2.2.1.6 Nounce

### 2.2.2 Ethereum

### 2.2.3 Decentralized Storage Providers

#### 2.2.3.1 IPFS

#### 2.2.3.2 StorJ

### 2.2.4 Metadisk: Blockchain-Based Decentralized File Storage Application

#### 2.2.4.1 Dat-data

#### 2.2.4.2 Sia

### 2.2.5 Access control

# Chapter 3

# Method

### 3.0.1 Data Structure for Smart Contract

| Information to Store | Types in Solidity |
|---|---|
| Storage Provider | String |
| Status | String |
| Name of the Document | String |
| Hash | bytes32 (SHA 256) |
| Signature | bytes32 (Pub Key) |
| Metadata [Date,author] | [unit (unix time), String] |
| Content Hash | bytes32 |
| Verification of valid copy | ??? |

### 3.0.2 EBNF/BNF

```
<urn>::= <prefix>:<provider>
<prefix>::= urn:x-docs
<provider>::= <storj>|<ipfs>|<azure>|<amazon>|<GDrive>|<Dropbox>|<OwnCloud>

<storj>::= ???
<ipfs>::= ipns:<DHTHash>:<FilePath>
<azure>::= ???
<amazon>::= arn:aws:s3:::mybucket/photo.png
<GDrive>::= ???
<Dropbox>::= ???
<OwnCloud>::= ???
```

### 3.0.3 URN/RFC

2 ways to register: formal ID:
are assigned by the IETF consensus / reviewed by a lot of people to be standartized / takes a long time informal ID:
are assigned with a number as an identifier (eg: "urn-¡number¿")
takes 2 weeks disccusion after the sending of the registration paper (urn-nid@apps.ietf.org)

use X- (eg: usn:x-docs:)

# Chapter 4

# Results

# Chapter 5

# Conclusion and Future Work

# List of Figures

# List of Tables