Implementation of a Reputation System for CA Trust Management

Implementierung eines Reputationssystems zur Vertrauensverwaltung für CAs
Bachelor-Thesis von Isabella Dix
Tag der Einreichung:

1. Gutachten: Prof. J. Buchmann
2. Gutachten: Dr. J. Braun

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Informatik
CDC

Implementation of a Reputation System for CA Trust Management
Implementierung eines Reputationssystems zur Vertrauensverwaltung für CAs

Vorgelegte Bachelor-Thesis von Isabella Dix

1. Gutachten: Prof. J. Buchmann
2. Gutachten: Dr. J. Braun

Tag der Einreichung:

Contents

1. Introduction **2**

2. Background **4**
2.1. Information Technology Security 4
2.2. Attacks 4
2.3. Symmetric and Public Key Cryptography 5
2.4. Web Public Key Infrastructure 6
2.5. Computational Trust 11
2.6. CA - TMS 13

3. Extensions for CA-TMS - Service Providers **15**
3.1. General 15
3.2. Reputation System 20
3.3. Pushes 23
3.4. Evaluation 26

4. Conclusion **30**
4.1. Implementation 30
4.2. Evaluation 30
4.3. Open topics and future work 30

Abbreviations **31**

Bibliography **32**

A. Appendix **36**
A.1. Statement 36

1 Introduction

Communication over the internet makes up a significant part of our every day life. Not only in private, but also in a business environment. The monthly traffic sent over the internet is estimated to be about 88 exabyte of data in 2016. And the number is continuously growing. For the year 2019 the number is estimated to be nearly twice as high (168 exabyte) [1]. In 2014 a study on global internet usage was conducted. It showed that apart from passive actions like ”exploring new subjects” or ”find locations when walking or driving around” a majority of users stated to go online to ”connect with family and friends” (73%) or ”access products and services and make purchases” (92%) [2]. Both activities yield the risk to leak private or sensitive data to unauthorized persons. This data can be private messages in a social network as well as payment information for e-commerce or online banking credentials. It is therefore fundamental that these communications are secure.

Secure connection usually require three matters: Authentication, which guarantees that the communicating entities can be sure that the other entities really are, who they pretend to be. Integrity, which ensures that the communication's contents have not been changed on their way from one entity to another. And confidentiality, which provides that only the known parties can access the contents of this communication.

To achieve such secure connections on a insecure networks such as the internet, an architecture called the Web Public Key Infrastructure (Web PKI) was established. We will address the details in the next chapter and just give a conceptual and simplified overview here. An integral part of the Web PKI are special third parties which attest certain characteristics for a website. If a user visiting the website trusts the third party, he believes that the website has the characteristics that were certified by the it. Especially does he have to believe in these characteristics in order to be able to establish a secure connection to the website.

The problem now is, that the average user does not know this design and can not be expected to comprehend or handle issues arising when such a secure connection could not be established, because e.g. the third party is unknown. This issue was addressed by including a list of trusted third parties to all standard browsers currently in use. Every third party contained in this list is then automatically trusted by the user. The list has to be large enough to assume that the user will not encounter a website, that was not directly or indirectly certified by a third party contained in the stored list.

The problem with this design is: The bigger the list, the easier it is possible for adversaries to attack. An attacker only has to find one weak link to obtain an erroneous attestation of the characteristics mentioned above. The user will thus be under the impression, that the adversaries website is trustworthy and can be lead to expose sensitive data.

One of the notions to secure the Web PKI therefore is to minimize the list of trusted third parties, as far as possible and sensible. This is the objective of a system called Certification Authority - Trust Management System (CA - TMS), which was introduced in [13]. The parts described in [13], chapter 4 were already implemented, while this thesis will cover the details to the implementation of [13], chapter 5.

The subsequent chapters are organised as follows: First we give some background explaining the schemes used in this thesis. This will especially include the concepts of encryption, digital signatures, the Web PKI, computational trust, as well as an overview over the already implemented system. Afterwards we focus on the realisation of the system's extensions implemented in this thesis and give an evaluation over the used methods. Finally we conclude the thesis and discuss future work.

2 Background

In this chapter background information necessary to follow the topic of this thesis is presented. It starts with a brief overview over the general objectives in Information Technology Security (IT security), followed by the attacks referenced in this thesis, concepts of Public Key Cryptography and an outline of the Web PKI, its vulnerabilities and the suggested techniques to mitigate these threats. Subsequently, a summary over the concept of computational trust is given, as well as an introduction to the already implemented client application, the CA Trust Management System and its features.

2.1 Information Technology Security

The primary objectives in information technology security are availability, integrity, confidentiality, as well as authentication and non-repudiation. Availability refers to a system's characteristic to ensure that an asset (hard- or software) can be used by any authorized entity whenever requested. Integrity denotes the property that an asset is only modified by authorized parties. Confidentiality expresses the ability of a system to assure that data is viewed only by authorized entities. Authentication is the process of confirming an entity's identity. Non-repudiation refers to the property that an entity cannot successfully deny having performed an action [3].

2.2 Attacks

This section covers some well-known attacks that are relevant to this thesis.

2.2.1 Man in the middle attack

Man in the middle (MitM) depicts an attack where an adversary intercepts a communication between two parties A and B without their knowledge. In its attacker model it is assumed that the attacker can impersonate any of the two entities A and B to each other. He is such able to eavesdrop or alter the communication to his content [4].

2.2.2 Sybil attack

A sybil attack is an attack which infiltrates a peer-to-peer-network with fake identities. It usually aims at reducing the reliability or effectiveness of the network, by letting the forged identities propagate false information [5].

2.2.3 Denial-of-service attack

A Denial-of-service (DoS) attack is an attempt to make a service or resource unavailable to users by spamming it with requests such that the high load makes it impossible to process other requests. A well-known version of this is the Distributed-Denial-of-Service (DDoS) attack. It sends the requests from different computers so that not only tracing is less feasible, but also that many more requests can be sent at the same time [4].

2.2.4 Structured Query Language-Injection

Structured Query Language (SQL) is a programming language that provides standardized schemes to manage databases [6]. If an attacker knows the database scheme in a web-application, he can try to insert malicious code e.g. to obtain or edit private data stored in the database. This can be realised by inserting SQL code in unprotected input fields provided by the system (e.g. fields in a registration form) [7].

2.2.5 Domain Name System Spoofing

In order to map a domain name (e.g. "https://www.verisign.com/") to an internet protocol (IP) address, that is used on the lower network levels, the so called Domain Name System (DNS) is used. If an attacker can add or alter entries in the DNS, he is able to redirect users to malicious websites that they consider trustworthy. Such an attack is called DNS spoofing [8].

2.3 Symmetric and Public Key Cryptography

The objectives mentioned in 2.1 are commonly achieved by the use of symmetric and public key cryptography. In the following an overview over the most important cryptographic components and schemes is given.

2.3.1 Symmetric key encryption

In order to communicate confidentially, messages between two parties are sent encrypted in such a way that only the two authorized entities can decrypt them. For this purpose the two entities must know a symmetric key, that can be used for en- and decryption of the data. Well known encryption algorithms include the Triple-Data Encryption Standard and the Advanced Encryption Standard (AES) or Rijndael [9].

2.3.2 Hash functions

Hash functions are functions $h : \{0,1\}^* \mapsto \{0,1\}^n$. They map strings of arbitrary length to strings of a denoted length $n \in \mathbb{N}$. Two values $x_0, x_1 \in \{0,1\}^*$ that result in the same hash value $h(x_0) = h(x_1)$ are called a collision. In order to consider a hash function as secure, it needs to be a one-way function. Meaning that the function is easily computable, but practically impossible to revert. Moreover must it be difficult to find collisions. For more detailed information and formal definitions we refer to [10]. Well known and currently considered secure hash functions include SHA-1, RIPEMD-128 and RIPEMD-160. MD4 and MD5 are today considered insecure [11].

2.3.3 Digital Signatures

Digital signatures generally serve the same purpose as physical (written) signatures. They verify that a person has signed a document, meaning e.g. that its content has been taken note of or

that the signer verifies the correctness of its content. However, in contrary to physical signatures, it has to be ensured that a signature can not be copied. If an entity wants to create a digital signature, it needs a private key d, that is only known to this entity and an associated public key e, that can be accessed and used by any party to verify the signature. The private key must not be derivable from the public key [9].

2.3.4 Public key encryption

Analogous to digital signatures, but in reversed fashion, public key encryption denotes a cryptographic scheme where the sender encrypts messages with the receiver's public key e. The receiver needs his private key d in order to be able to decrypt messages [9].
Well known encryption algorithms include the Rivest-Shamir-Adleman Algorithm (RSA Algorithm), Rabin and ElGamal cryptosystem [11].
Just like with digital signatures a way needs to be found to publish the public key in such a way that the affiliation between the key and its owner can be verified by any other entity [9].

2.4 Web Public Key Infrastructure

In the past many architectures for publishing and distributing public keys have been suggested. The most common of these nowadays is the Web PKI.

General

Here the belonging of a public key to its owner is certified by a certificate digitally signed by a trustworthy third party. Such a party is called a certification authority (CA). The owner of the public key is called the certificate's subject, the CA its issuer. CAs can under certain restrictions also sign certificates for other (Sub) CAs. Keys can also be validated by multiple certificates from different CAs. Often the CA maintains a subordinate registration authority (RA), which is responsible for verifying the subject's identity before issuing the certificate. The used certificates are standardized by X.509 and therefore are often referred to as X.509 certificates [12].

Structure

The architecture is hierarchical, meaning that if an entity trusts a CA, it trusts all Sub CAs, for which the CA issued a certificate as well. The structure is not strictly centralized, but rather organized in sets of so called Root CAs, which provide the roots for multiple certification trees. Root CAs sign their certificates themselves (issuer and subject are the same), which gives the certificates the name self-signed certificate [12].
An example for such a certification tree can be seen in Figure 2.1. The boxes represent different entities and the arrows depict certificates, where circular arrows stand for self-signed certificates. A path in such a tree is referred to as a certification path.

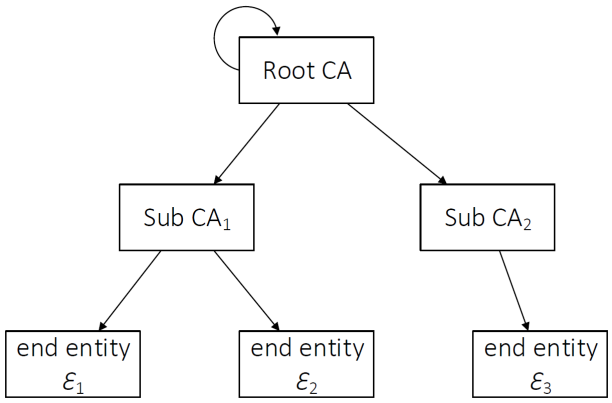

Figure 2.1.: Simplified sample PKI as taken from [13]

Revocation

Certificates have a defined life-span until they expire. Under certain circumstances they can be revoked before the end of their validity period, e.g. if the private key belonging to this certificate's public key has been compromised or if the private key's owner requests it. In order to check whether a certificate is still valid, so called certification revocation lists (CRLs) are used, which hold a list of revoked certificates for each CA. Alternatively, one can use the Online Certificate Status Protocol (OCSP). It is a protocol which takes a certificate and returns some information on its current status. Whether it is not revoked ('good'), revoked ('revoked') or whether no definitive answer could be given ('unknown'). Note that 'good' must not yet give any information about the validity of the certificate in terms of expiration or even issuance. The exact extend on what these answers (especially the answer 'good') can mean in an implementation is left open to the developer [12].

Policies

So called certificate policies regulate the CA's duties and organizational concerns. These include among others information about when, how and how frequently certificates, revocation and policy information should be published, how identification and authorization is done, the application process for a certificate and its revocation. Moreover they include specifications on [12]:

- facility and management security
- hardware security in terms of secure key management
- CRL and OCSP management
- compliance audits and assessments
- other business and legal matters

Trusted Lists

To be able to communicate encrypted with a website a user needs to trust the website's certificate or a certificate that lies upside in the certification path.This must be the case for every

connection that the client requires to be confidential (e.g. every online shopping site). In order to increase client compliance - most importantly with non-professionals - so called trusted lists are incorporated in every standard internet browser today. These trusted lists contain all certificates which this internet browser trusts automatically. Meaning that all certificates which are part of the certification trees spanned by the certificates in this list are automatically considered trustworthy [12].

Protocols

Today's standard for encrypted communication is the Transport Layer Security (TLS) Protocol, which is the successor and based on Security Sockets Layer (SSL). It provides means for validating certificates for both communication sides (client and server), extracting the key to be used for encryption and maintaining a session between two communicating entities. This process is mostly referred to as SSL handshake. The protocols TLS and SSL are mostly used in combination with the Hypertext Transfer Protocol (HTTP), thus it is also referred to as HTTPS [14].

2.4.1 Issues

Several issues with this structure have been detected over the past years. But because of its global nature, very few of them could be addressed effectively ([15], [16], [17]). An overview over the most important ones will be given in the following.

Impersonation

An attacker who can disguise himself as a trusted entity is able to perform a MitM attack. For example this can happen when the attacker is able to spoof the RA's identification process and gets issued a certificate, although usually the RA would not consider him trustworthy. Another scenario in which this is possible is when the attacker can compromise the RA itself, skip the authorization process and proclaim himself trustworthy on behalf of the RA. Many of such cases have been brought to light in recent years, although security experts claim a much higher number, that was never disclosed to the public [13].

CA compromise

If an attacker manages to obtain CA system access or simply the attacked CA's (private) signing key he can issue certificates for arbitrary and potentially malicious entities. Due to the trusted lists the browser will declare all certificates trustworthy until the CA realizes the compromise and revokes its certificate. Unfortunately not all compromises are disclosed to the public due to the too-big-to-fail problem (e.g. *Verisign* inter alia operates the authoritative registry for all *.com* top-level domains [18]) [13].

Trusted lists

As already established to provide interoperability, the trusted list has to include all Root CAs whose certification trees span all those certificates which are likely to be used by the user. Over

the years these lists have become increasingly large, containing more than 150 CAs in the case of Mozilla Firefox [19] and about 264 in Microsoft's root store [15]. While this is of course very useful for users who do not want to be disturbed while browsing, it brings several major drawbacks in case a CA becomes untrustworthy (e.g. because its key has been compromised) [16].
Problems lie not only in the fact, that security in this model is only as strong as the weakest link [17], but also that the major part of these white-listed CAs are irrelevant to a single user and therefore pose an unnecessary security threat [16]. Another issue is that many of the procedures done are not done transparently. Governments can compel any domestic CA to issue a certificate for any website. That way if the CA is part of the above mentioned pre-established white list, the browser will trust it and possibly expose private data indirectly to a number of governments (foreign or domestic) without knowing it. Thus making MitM attacks easily possible [15].

2.4.2 Security Analysis

In the following we present the general security and attack model used in this thesis.

Security model

The objective of Web PKI is to establish secure connections over an insecure network (the internet). By secure we mean a connection where at least one of the communicating parties was authenticated by the other and which provides authenticity, integrity and confidentiality. In general these connections are established between a web browser usually operated by a user and a web server, where the server authenticates itself by presenting a public key certificate. During the so called SSL handshake the client verifies the certificate and both parties establish a symmetric key to be used for communication [20].
The security model considers two entities ε_1 and ε_2. ε_1 wants to establish a secure connection to ε_2, where ε_2 has to be authenticated. Authentication should be given by a certificate C with public key $\boldsymbol{pk}$ and subject ε_2. In order to trust ε_2 the certificate has to be valid and ε_1 has to trust its issuer. These requirements can be verified by performing standard path validation on the certification path $p = (C_1, ..., C_n)$ where $C_n = C$ and the trust validation described in 2.6. Trust validation is not incorporated in the standard Web PKI. Instead trustworthiness is simply assumed to be the case for every participant equally [13].

Attacker model

Although we generally consider TLS connections, the focus is not on possible attacks on TLS protocol or handshake, but rather on attackers exploiting deficiencies in the Web PKI architecture. An attacker $\boldsymbol{A}$ generally aim at breaking authenticity, integrity or confidentiality.
In particular we consider attackers who aim at impersonating ε_2 without being detected. Impersonation can mean completely intercepting the communication so that ε_2 is never even contacted or acting as man in the middle between ε_1 and ε_2. For this purpose the attacker presents a fraudulent certificate, that ε_1 considers to be ε_2's and whose private key is available to the attacker. The concept can be seen in figure 2.2

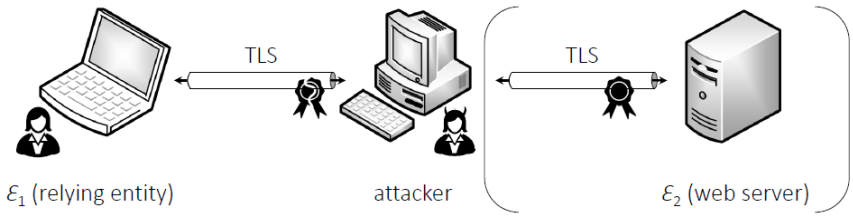

Figure 2.2.: Attacker impersonating ε_2 as taken from [13]

A is able to intercept the communication by performing DNS spoofing and receiving valid certificates from a CA of the Web PKI with a subject chosen by the attack. We call such certificates fraudulent as opposed to legitimate ones, where the subject can not be chosen but is actually the entity that controls the certificate's public key. Fraudulent certificates may be obtained by impersonating the subject or compromising the CA (see 2.4.1 for details).
We limit ***A*** to only one fraudulent certificate per CA at the same time. As soon as he chooses the CA, it is fixed. This is considered realistic, as the issuance of a fraudulent certificate is not trivial. Furthermore we consider both ε_1 and ε_2's systems secure, meaning that it is impossible for any attacker to alter or access locally stored data. ε_2's private key (belonging to the legitimate certificate's public key) can not be accessed by any attacker. And ***A*** is neither capable of breaking the cryptographic algorithms nor the secure connection. Therefore a secure connection once established between ε_1 and ε_2 cannot be broken by an attacker without being detected [13].

2.4.3 Techniques of Mitigation

A number of techniques have been proposed to reduce the points of attack. Some focus on improving the given architecture, some suggest new approaches. We first consider alternative architectures and then the approaches to secure the given PKI. Lastly we give an overview over which of these techniques were considered in the given approach.
Alternative architectures include Simple Distributed Security Infrastructure, a system derived from the Simple Public Key Infrastructure. They both propose that a certificate, while still binding key-pair and owner, does not yet give any warranties about the key holder is. The "relying" entity is forced to develop a judgement on its own. Another approach suggests the usage of reputation systems, that provide generally held positive opinions about an entity, which can be seen as a brand and that the entity can pass onto other entities. Moreover suggestions were made to focus more on privileges and restrictions than the identification of individuals, which leads to more privacy. And lastly one of the more influential approach is the so called Pretty Good Privacy (PGP) web of trust. For an entity ε_2 to be trusted by another entity ε_1 it requires a certain number of other entities ε_i that ε_1 trusts to trust in ε_2. Trust can be restricted or expanded by placing trust in every other entity in how far they are considered reliable to suggest trust for another entity [21].
One approach to further secure PKI is certificate pinning, the process of saving the key of a host's certificate when the host is visited the first time and verifying it on every revisit [22]. Another strategy is to limit the number of trusted CAs. The first notion of establishing a country-based trust was applicable only in the United States of America, due to the dominance of USA CAs [15]. A system was therefore suggested to limit the number of trusted CAs according to the

user's specific needs. It was shown that in doing so, the attack surface could be reduced by more than 90% [16]. This is also the approach considered in this thesis.

2.5 Computational Trust

In order to model trust placed in CAs, a design to characterize trust is needed. It should especially be able to show the degree of (un)certainty with which a decision was taken in order to make transparent how reliable this information is.
In literature the two commonly used definitions of trust are reliability trust and decision trust. Reliability trust depicts an individual's trust in another individual to perform an expected action, without being able to actually monitor if this action was done [23]. Decision trust is defined by Jøsang et al. in [24] (inspired by [25]) as the extend to which an entity is willing to depend on another entity in a given situation with a feeling of relative security, even though negative consequences are possible.
These definitions can help develop schemes to model and compute trust opinions. One such scheme will be presented in the following section.

2.5.1 CertainTrust and CertainLogic

Ries et al. have suggested a scheme called CertainTrust, which additional to the trust modelling brings along CertainLogic - a set of operators to combine CertainTrust opinions [26].

CertainTrust

An opinion o_A on a statement in CertainTrust is modelled as $o_A = (t, c, f)$. $t \in [0, 1]$ represents the average rating - meaning the opinion formed about the statement, where $t = 0$ would mean that the statement is believed to be wrong and $t = 1$ would mean the statement is believed to be correct. $c \in [0, 1]$ depicts the certainty with which we trust in t with respect to future decisions, where $c = 0$ would mean that we have not yet found neither evidence supporting nor contradicting the statement and $c = 1$ would mean that we consider the opinion given in t to be representative for future decisions. $f \in]0, 1[$ describes the initial expectation value, which is the assumption about the trust of a statement without any evidence. For soundness it was defined that t in this case should be 0.5 [26].

CertainLogic

Opinions can be combined using logical operators similar to propositional logic. Definitions for AND, OR and NOT have been defined in figure 2.3.
AND and OR are commutative ($o_{A \wedge B} = o_A \wedge o_B$ and $o_{A \vee B} = o_A \vee o_B$) and associative ($o_{A \wedge (B \wedge C)} = o_{(A \wedge B) \wedge C}$ and $o_{A \vee (B \vee C)} = o_{(A \vee B) \vee C}$), but not distributive ($o_{A \wedge (B \vee C)} \neq o_{(A \wedge B) \vee (A \wedge C)}$) [26].
Additional to usual logic operators a new fusion operator was introduced. It provides means to merge multiple trust information if the individual information can not be considered independently e.g. because preferential weighting is required. The authors of [27] proposed three fusion operators: aFusion, wFusion and cFusion. For this thesis only cFusion will be used, it will

OR	$c_{A\vee B} = c_A + c_B - c_A c_B - \frac{c_A(1-c_B)f_B(1-t_A) + (1-c_A)c_B f_A(1-t_B)}{f_A + f_B - f_A f_B}$ $t_{A\vee B} = \begin{cases} \frac{1}{c_{A\vee B}}\left(c_A t_A + c_B t_B - c_A c_B t_A t_B\right) & \text{if } c_{A\vee B} \neq 0\ , \\ 0.5 & \text{else}\ . \end{cases}$ $f_{A\vee B} = f_A + f_B - f_A f_B$
AND	$c_{A\wedge B} = c_A + c_B - c_A c_B - \frac{(1-c_A)\,c_B\,(1-f_A)\,t_B + c_A\,(1-c_B)\,(1-f_B)\,t_A}{1-f_A f_B}$ $t_{A\wedge B} = \begin{cases} \frac{1}{c_{A\wedge B}}\left(c_A c_B t_A t_B + \frac{c_A(1-c_B)(1-f_A)f_B t_A + (1-c_A)c_B f_A(1-f_B)t_B}{1-f_A f_B}\right) & \text{if } c_{A\wedge B} \neq 0, \\ 0.5 & \text{else}\ . \end{cases}$ $f_{A\wedge B} = f_A f_B$
NOT	$t_{\neg A} = 1 - t_A$, $c_{\neg A} = c_A$, and $f_{\neg A} = 1 - f_A$

Figure 2.3.: The operators as taken from and defined in [26]

therefore be the only one described here.
cFusion aims at reflecting conflicts in the result, for example when one entity is very certain about the correctness of a statement, while the other is very certain about the falsehood thereof. The result will mirror this difference in lowering the certainty of the result. Furthermore cFusion can take weighting factors into account to give higher or lower importance to the different opinions.
Let A be a statement and let $\{o_{A_i} = (t_{A_i}, c_{A_i}, f_{A_i}) \mid \quad 1 \leq i \leq n\}$ be n opinions about A. Additionally assign a weight w_i to every opinion o_{A_i}, where $w_i \in \mathbb{R}_0^+$ and $\sum_{i=1}^n w_i \neq 0$. Then the cFusion of these opinions with the respective weights is defined as:

$$\widehat{\bigoplus}_c(o_{A_1}, o_{A_2}, \ldots, o_{A_n}) = (t_{\widehat{\oplus}_c(A_1,A_2,\ldots,A_n)},\ c_{\widehat{\oplus}_c(A_1,A_2,\ldots,A_n)},\ f_{\widehat{\oplus}_c(A_1,A_2,\ldots,A_n)})$$

$$t_{\widehat{\oplus}_c(A_1,A_2,\ldots,A_n)} = \begin{cases} \frac{\sum_{i=1}^n w_i t_{A_i}}{\sum_{i=1}^n w_i} & \text{if } c_{A_i} = 0 \\ 0.5 & \text{if } c_{A_i} = 1 \\ \frac{\sum_{i=1}^n (c_{A_i} t_{A_i} w_i \prod_{j=1, j\neq i}^n (1-c_{A_j}))}{\sum_{i=1}^n (c_{A_i} w_i \prod_{j=1, j\neq i}^n (1-c_{A_j}))} & \text{if } \{c_{A_i}, c_{A_j}\} \neq 1 \end{cases}$$

$$c_{\widehat{\oplus}_c(A_1,A_2,\ldots,A_n)} = \begin{cases} 1 - DoC & \text{if } c_{A_i} = 1 \\ \frac{\sum_{i=1}^n (c_{A_i} w_i \prod_{j=1, j\neq i}^n (1-c_{A_j}))}{\sum_{i=1}^n (w_i \prod_{j=1, j\neq i}^n (1-c_{A_j}))} \cdot (1 - DoC) & \text{if } \{c_{A_i}, c_{A_j}\} \neq 1 \end{cases}$$

$$f_{\widehat{\oplus}_c(A_1,A_2,\ldots,A_n)} = \frac{\sum_{i=1}^n w_i f_{A_i}}{\sum_{i=1}^n w_i}$$

$$DoC = \frac{\sum_{i=1,j=1}^n DoC_{A_i,A_j}}{\frac{n(n-1)}{2}}, \qquad DoC_{A_i,A_j} = |t_{A_i} - t_{A_j}| \cdot c_{A_i} c_{A_j} \cdot \left(1 - |\frac{w_i - w_j}{w_i + w_j}|\right)$$

The cFusion operator is commutative [27].

2.6 CA - TMS

The CA - TMS is a Java application communicating with a Firefox Browser Plugin, which aims at reducing the number of automatically trusted CAs, while extending the validation processes. It includes functionalities for extended trust validation, pinning, revocation checking, monitoring certificates over a longer period of time and using external notary services.
For this purpose it keeps a so called trust view **View** as introduced in [28]. The trust view saves trust assessments for certificates (the exact denotation of what an assessment is, will be given later in this chapter), the certificates themselves (trusted or untrusted) and a watchlist of certificates for which no definitive ruling could be given. The path validation on occurrence of a TLS connection to a website is performed by Firefox and then given to the Java application for extended validation as described later in this chapter. Therefore the certificates relevant for this are the ones in Firefox's Root Certificate Store and not the ones in the Java Runtime Certificate Store [29].
In order to understand the changes brought by the addition of service providers (SPs), the trust assessments and the essentials extracts from the algorithms for update of the trust view and validation will be introduced shortly.

Trust assessments

A trust assessment is a tuple $(\boldsymbol{pk}, \boldsymbol{CA}, S, o_{kl}, o_{it}^{ca}, o_{it}^{ee})$. It depicts an assessment on the trust for a certificate with public key $\boldsymbol{pk}$ and name of the CA (issuer) $\boldsymbol{CA}$. S holds a set of certificates with this CA and public key, that have previously been verified by this user. o_{kl} represents the user's opinion on whether this public key really belongs to the CA or not, it is called key legitimacy. To model how much the user trusts the CA to be reliable when issuing CA certificates o_{it}^{ca} is an opinion object (issuer trust for CA certificates). As is o_{it}^{ee}, which depicts how much the user trusts the CA to be reliable when issuing end entity certificates (issuer trust for end entity certificates) [13].

Validation algorithm

The validation algorithm is given the following inputs: the certification path $p = (C_1, ..., C_n)$ without intermediary self-signed certificates, the user's trust view **View**, the security level $l \in [0, 1]$, a list of validation services, which are out of this thesis's scope and (optionally) a SP. The algorithm should output some $R \in \{trusted,\ untrusted,\ unknown\}$ and proceeds as follows [13]:

1. if C_n is already saved as trusted in **View**, set $R \leftarrow\ trusted$
2. if p contains a certificate that is saved as untrusted in **View**, set $R \leftarrow\ untrusted$
3. if C_n is not contained in **View**
   a-k) calculate R (ommited)
   l) if ($R =$ untrusted) report p to **SP**
4. return R

This part of the algorithm will be relevant in section 3.3.

TrustView Update algorithm

When the trust view needs to be updated with new trust assessments, the following inputs are required: A certification path $p = (C_1, ..., C_n)$ without intermediary self-signed certificates, the user's trust view **View**, the output $\boldsymbol{R}$ of the trust validation, a list of new assessments **TL**, a boolean value $v \in \{\text{true}, \text{false}\}$, which shows whether C_n was validated successfully, a list of validation services VS, which is not interesting for this thesis and optionally a reputation system **RS**, which on input of a pair $(\boldsymbol{pk}, \boldsymbol{CA})$ outputs unknown or a recommendation for the issuer trust $\boldsymbol{RS}(\boldsymbol{pk}, \boldsymbol{CA}) = (\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$.
After termination the trust view should be updated with the new trust assessments.
In (the optional) step 3 the functionality of the SP is defined as follows [13]:

3. if ($\boldsymbol{R} =$ trusted) and $v = true$, then for all $TA_i \in TL$ do
   (a) request $\boldsymbol{RS}(\boldsymbol{pk}_i, \boldsymbol{CA}_i) = (\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$ from **RS**
   (b) if $\boldsymbol{RS}(\boldsymbol{pk}_i, \boldsymbol{CA}_i) \neq$ unknown
      if $(i < n-1)$ set $i_{it,i}^{ca} = (0.5, 0, E(\tilde{o}_{it,i}^{ca}))$
      else set $i_{it,i}^{ee} = (0.5, 0, E(\tilde{o}_{it,i}^{ee}))$

This part of the algorithm will be relevant in section 3.2.

3 Extensions for CA-TMS - Service Providers

This chapter describes the extension of CA-TMS by SPs their components and implementation. We start with an overview over the components and their purpose as well as some basic architectural decisions in 3.1. Afterwards we describe the two main features and their implementation in 3.2 and 3.3. Finally, an evaluation is given in 3.4.
Some functionalities have not yet been put into practice. To provide a thorough survey over the complete system they are included in the respective chapter, but are named separately in an additional last sub-chapter.

3.1 General

The existing CA-TMS application is to be extended by the concept of SPs as described in [13], chapter 5. A SP is a server, where clients can register, upload their trust views and get suggestions and warnings for future trust decisions. Two main features have been described.
The existing system already brings along some functionality for extended validation in case of incomplete local information. This is now extended by the reputation system, which represents an additional external system to increase the amount of information for future decisions. It will be described in section 3.2.
One of the PKI's main defects is the possibility of entities trusting a careless, compromised or simply malicious CA, which knowing- or unknowingly issues certificates for fraudulent subjects. Often enough cases of compromise are not disclosed to the public [30], [31]. In order to speed up the process of information propagation in case an entity discovers a behavioural change in a previously trusted CA, [13] suggested a push-service where other relying entities can be warned about a possibly malicious CA and subsequently re-evaluate their prior assessment on the trust-worthiness of this CA. It will be described in section 3.3.

3.1.1 System model

For the following chapters the system model will be as follows:
There exists an entity $\boldsymbol{\varepsilon_1}$ with a trust view **View**, which wants to examine the trustworthiness of another entity $\boldsymbol{\varepsilon_2}$'s key. $\boldsymbol{\varepsilon_1}$ is registered at SP $\boldsymbol{SP_1}$ and so are other users $\boldsymbol{U_1}, ..., \boldsymbol{U_n}$. Moreover there exists a network of SPs $\boldsymbol{SP_1}, ..., \boldsymbol{SP_m}$. It is assumed that the SPs have pre-established trust relationships, which allow them to communicate securely. The network does not have to be complete, meaning that a SP does not have to know any other SP. Every user can choose for himself which SP he wants to be registered at. As soon as the server accepts the user's registration, the user can upload his trust view. Every SP will therefore have its own user base and trust views.
Figure 3.1 shows the model, figure 3.2 the SP's architecture and figure 3.3 the architecture of the CA TMS system (extended from the version in [29]).

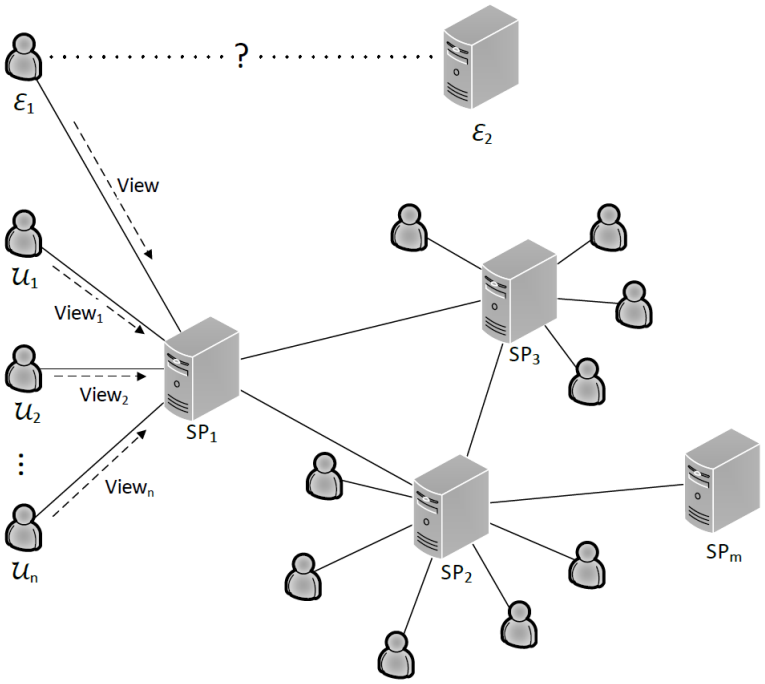

Figure 3.1.: The system model as taken from [13]

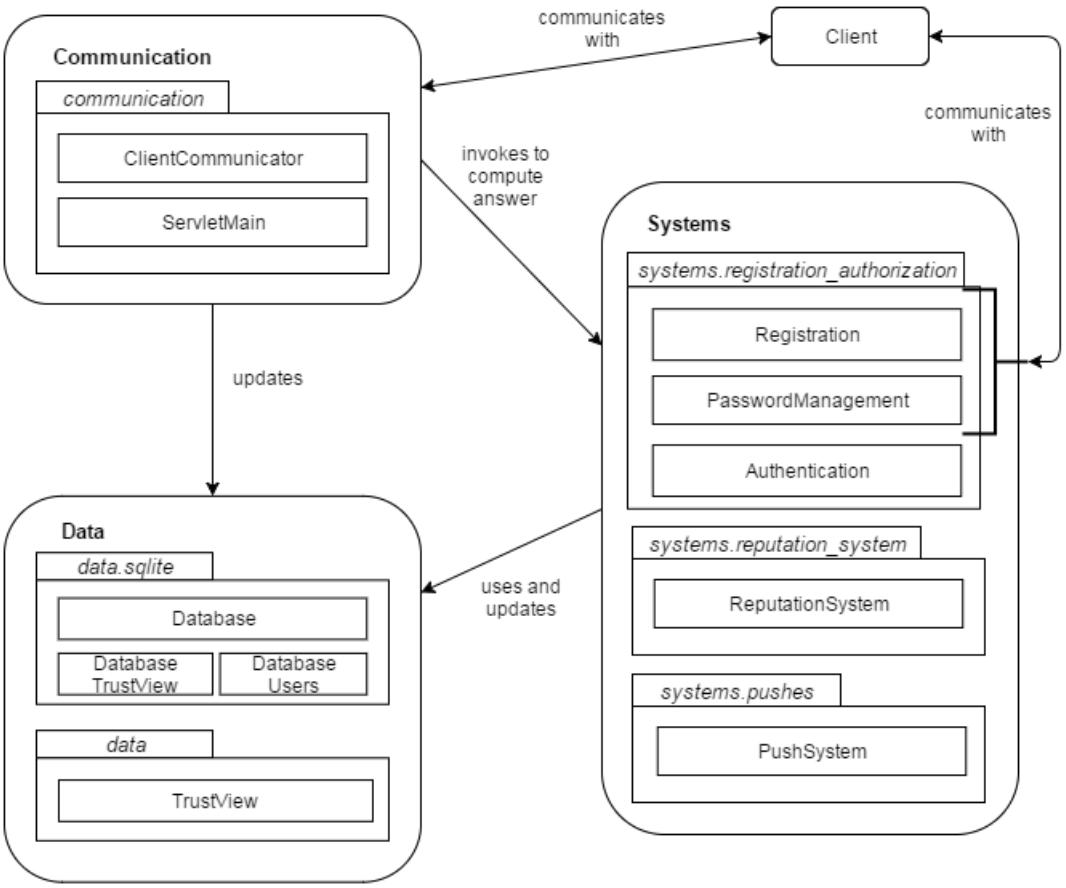

Figure 3.2.: The SP's architecture

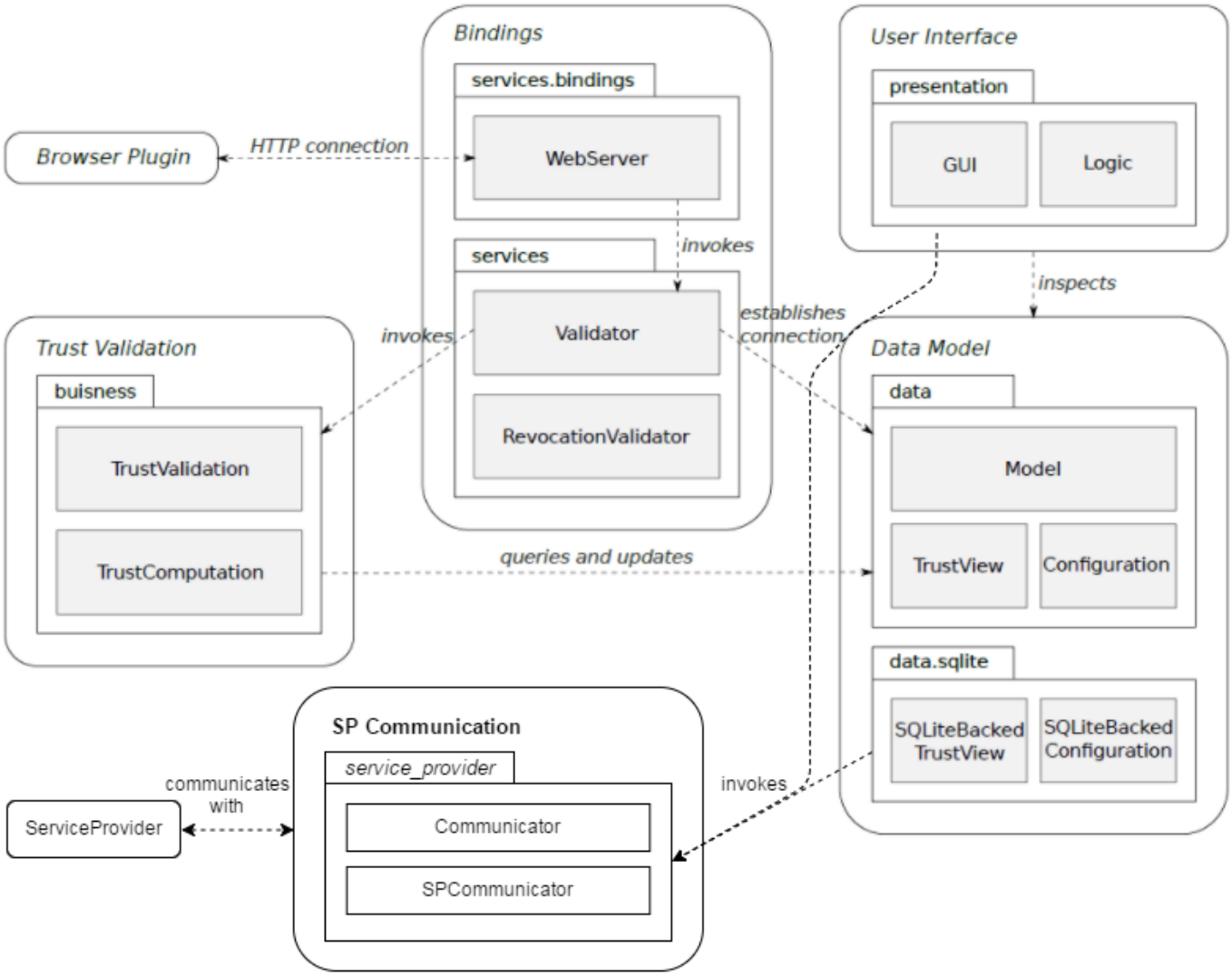

Figure 3.3.: The client's architecture, expanded version from [29]

3.1.2 Challenges

Through the implementation of this part of the system the following challenges had to be addressed.

Server Architecture

A general structure for the server should be chosen. There exist two basic architectures that in this case would make sense. The service oriented architecture (SOA) and the representation state transfer (REST), also referred to as RESTful Web Service.
SOA is generally described as an architecture in which software is constructed as a composite applications made up of services running on different nodes and communicating via message passing. It uses Web Service Description Language (WSDL) for service interfaces, simple object access protocol and extensible markup language (XML) for parsing and HTTP for sending messages [32]. To provide interoperability between different services an additional integration layer is introduced to specify the communication protocol between the single nodes [33].

REST on the other hand ignores the details of component implementation and protocol syntax in order to focus on the roles of components. It especially leaves its encoding and message parsing for the developer to choose. It only supports HTTP and is mostly considered stateless, meaning that e.g. session management must be done by the client [34].
For its more lightweight nature REST has been chosen for this thesis. Javascript Object Notation (JSON) has been used instead of XML for message parsing.

Server-Client-Communication

In order to communicate successfully server and client need some kind of protocol to understand each other. As already established they are connected via a TLS connection, which means that underlying HTTP is used. As both client and server are developed in co-dependence, it is possible to introduce two custom header fields for easier communication. This is not a standard practice and even discouraged when any of the parties should be able to communicate with arbitrary other parties, because errors could arise if required header fields are not found or understood. However in this case it provides a simplification of the message-parsing, as the information about the content of the HTTP message is already introduced in the header and can thus easier be extracted by the recipient. Contrary to earlier usages the fields are not distinguished from standardized fields by prefixes ”X-*” [35].
The field ”Action” holds the action that the SP should execute for the user, the field ”Detail” the details to this action. They are of following types:

”Action” $\in$ {add, remove, move_from_trusted_to_untrusted, move_from_untrusted_to_trusted, set_revoked, push_to_sp, push_from_sp, get_suggestion, test}

”Detail” $\in$ {assessment, certificate_trusted, certificate_untrusted, certificate, all, account, null}

With the combination of these two all required actions can be requested from the server.
The body of the HTTP-message then has to hold the information necessary to successfully execute these actions.
Additional to these headers the ”Authentication” header is used for (basic) authentication and the ”Cookie” header for storing sessions.

Client-Authentication

The client (CA-TMS) system has to be authenticated by the SP in order to have access to all services. There are several options for authentication, which will be examined for the given scenario.
Basic Authentication depicts the authentication scheme where every user has a username and associated password. He then can authenticate himself to the server by sending these credentials. The password and username are sent in plaintext (usually Base64-encoded). This would be a drawback, if the communication between server and client was not encrypted, which it is in this case. And although various attacks on TLS are known [36], the handling of this problem would exceed this thesis's topic [37].
Digest (Access) Authentication works in a similar way besides that the username and password

are hashed before sending. To prevent replay-attacks, it is constructed in a challenge-response-like way. On a request from a client the server sends a nonce, which the client integrates in its hash of username, password, HTTP method and requested URI and then transmits it to the server with the username. This way the password is never sent in clear text. Problems lie in its standardized use of MD5 as hash-function, which for a long time has been known as insecure [38], [37].
Another possibility is to integrate some hard-coded token/fingerprint into the client which can then be used to authenticate at the server. This is no option, because pass-phrases should never be integrated into code, especially not as this software is intended to be released as open source.
And there are also different versions of certificate-based authentication, which have the advantage of additionally verifying the user's identity - either by a trusted third party or by the server itself. However, this requires a lot of overhead, not only in verifying the user's identity, but also for the SSL-handshake in terms of performance. As the only reason for verifying the user's identity would be to reduce the attack surface for sybil-attacks, which is already done by appropriate selection of the recommender's trust views (see Section 3.2.2), the performance/overhead-concerns outweigh the advantages.
Taken all things into consideration, the choice has fallen on basic authentication via TLS, because it brings the basic security needed while being very ressource-friendly (in terms of time and storage-space).

Protection of passwords

Two new passwords have been introduced.
For authentication on the SP the user needs a password, which we will refer to as the authentication-password. It must be re-storable for the system in order to communicate with the SP, but at the same time may not be saved unencrypted due to security reasons. In order to achieve the first the authentication-password needs to be saved in such a way that it can be read by the system, meaning that it may not be hashed. For security reasons it may not be unencrypted. It therefore would have to be saved in plain-text or encrypted by another password, which we well refer to as the master-password.
The master-password is the password that is used for encryption. Not only of the authentication-password, but also later for the end-certificates.
The authentication-password is therefore saved encrypted in the user's database and can be decrypted on input of the master-password. On the SP's side it is saved hashed and salted, so that it can not be restored, but only be used for authentication on input of the user's request.
The master-password is only saved in the user's database and has to be queried from the user on every usage. However to bypass this, the user can let the system save the password in a variable until the next session. While this facilitates the handling for the user, it does avoid to save the password in code, which would not be advisable.
AES-128 (AES with 128b block- and key-length) was used for encryption, and SHA-1 with PBKDF2 for hashing and salting the password. For latter we used code from [39].

Other security issues

The server has been protected against SQL-injections by using prepared statements. Some obstruction against sybil-attacks could be gained by adding a captcha to the registration form as suggested by [13]. Although this can not fully prevent sybil attacks, it greatly mitigates this threat, by making it significantly more difficult to create an account. The account-creation now requires a human to participate, which takes longer than automatically (e.g. by a script) filling the form fields. reCAPTCHA v2.0 by Google (see more here [40]) was used.

3.1.3 Outlook

This chapter holds functionalities, which have not yet been implemented, together with some considerations on the realisation.

Communication between SPs

The SPs should have a pre-established trust-relationship, meaning that one SP knows and trusts a number of others on first start up. These SPs can be added manually to the database. In order to establish a TLS connection between two SPs, two options for authentication exist. One solutions is that the usual one-way authentication verifies the contacted SP's integrity and this SP then has to authenticate the contacting SP manually. Another option would be to use the both-way authentication [14], which authenticates both parties. A mixture of both is possible as well.

Further securing the registration process

Further obstruction against sybil attacks could be gained by modifying the registration process. For example we can solely accept email addresses as usernames and respond to the registration with an email. The registration process would then be extended as follows: In order to complete the registration, the user has to click a link in this mail, which confirms to the server that this email address really belongs to the user. Only then will the account be fully functional.
As already mentioned in the context of the added captcha, this can not fully prevent sybil attacks, but make them more expensive. It calls for multiple email addresses for multiple accounts. Although these can be set up or even be used from a single account (as described here [41]), this further obstructs an attacker from creating multiple accounts [13].

3.2 Reputation System

The Reputation System aims at making accumulated information available to different parties without disclosure of private data of the single users. Thereby providing relevant information to the requesting entity without revealing private data.

3.2.1 Functionality

The Reputation System extends the trust validation process in the following way. If an entity $\boldsymbol{\varepsilon_1}$ updates its trust view with a new assessment for a CA **CA** with public key **pk**, it requests a recommendation for the issuer trust. The server then gathers a list of trust views that are considered "similar" and accumulates them to a recommendation $(\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$. The exact meaning of similar trust views and how the accumulation is done, will be shown in section 3.2.2.
If no similar trust views could be found, the SP forwards the request to other SPs he knows, until he gets a recommendation to return to the user or **unknown**, if the other SPs could give no recommendation either. It should be transparent for a requesting entity that the results were derived from another SP. If it receives multiple recommendations (e.g. because he queries more than one other SP in parallel), they are aggregated using the cFUSION operator with equal weights.
After receiving the recommendation $\boldsymbol{\varepsilon_1}$ integrates it into his trust view **View**. The protocol has been formally described as follows [13]:

1. $\boldsymbol{\varepsilon_1}$ establishes a TLS connection to $\boldsymbol{SP_1}$ and authenticates itself
2. $\boldsymbol{\varepsilon_1}$ sends the pair (**pk**, **CA**) to $\boldsymbol{SP_1}$ using the secure connection
3. Depending on $\boldsymbol{View}$, $\boldsymbol{SP_1}$ selects $j \geq 0$ trust views $\boldsymbol{View_1}, ..., \boldsymbol{View_j}$ from its database (see Section 3.2.2 for details)
4. If $j > 0$:
   a) for $1 \leq i \leq j$ extract $o_{it,i}^{ca}$ and $o_{it,i}^{ee}$ for (**pk**, **CA**) from $\boldsymbol{View_i}$
   b) aggregate the opinions with the cFUSION operator: $\tilde{o}_{it}^{ca} = \widehat{\bigoplus}_c(o_{it,1}^{ca}, ..., o_{it,j}^{ca})$ and $\tilde{o}_{it}^{ee} = \widehat{\bigoplus}_c(o_{it,1}^{ee}, ..., o_{it,j}^{ee})$
5. If $j = 0$: $\boldsymbol{SP_1}$ forwards the request to other SPs it trusts, until it receives a recommendation $(\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$ or unknown from all. In order to enable the SPs to select the suitable trust views out of their databases, $\boldsymbol{SP_1}$ has to hand over $\boldsymbol{\varepsilon_1}$'s trust view. To maintain the user's privacy it is shortened by all end entity certificates beforehand.
6. $\boldsymbol{SP_1}$ responds to $\boldsymbol{\varepsilon_1}$ with either the aggregated issuer trust opinions $(\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$ or **unknown**
7. $\boldsymbol{\varepsilon_1}$ integrates the recommendation into **View**

3.2.2 Challenges

Through the implementation of this part of the system the following challenges had to be addressed.

Trust view selection and trust aggregation

On receiving a recommendation-request for (**pk**, **CA**) the server has to decide which trust views resemble the one requesting the recommendation. This is done because the recommendation should be based on the user's needs. Simply averaging all opinions may not suffice and even increase the threat of a sybil-attack (assuming that some of the registered users are malicious).

Similarity is measured with the Jacard Similarity Index (JSI).
The JSI measures the similarity of sets. For two sets A, B it is defined as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

In the case of trust views, we define two trust assessments as equal, if the **CA** and **pk** are identical. Therefore if n_i is the number of trust assessments in $\boldsymbol{View_i}$ and n is the number of assessments shared by $\boldsymbol{View_1}$ and $\boldsymbol{View_2}$, we can define the JSI for trust views as

$$J(\boldsymbol{View_1}, \boldsymbol{View_2}) = \frac{n}{n_1 + n_2 - n}$$

The JSI then indicates the similarity, which states the weight, with which this should weigh into the suggestion. The more similar, the higher the weight. In practice this works in the following manner: $\boldsymbol{SP_1}$ retrieves all trust views from its database that include a trust assessment for (**pk**, **CA**). For each of the found $\boldsymbol{View_i}$, it computes the weight $w_i = J(\boldsymbol{View}, \boldsymbol{View_i})$ and discards any $\boldsymbol{View_i}$ with $w_i \leq b$. Afterwards the issuer trust is aggregated using the weights w_i and the cFUSION-operator (see Section 2.5.1).
By considering the similarity between the trust views, we make the suggestion even more relevant to the user as issuer trusts from trust views similar to $\boldsymbol{\varepsilon_1}$'s have greater influence on the suggestion than those less similar. The pruning of those below a certain weight is done because the Jacard similarity only considers weights relatively, meaning that if only trust views with a very low weight are found, they will go into the suggestion with relatively high weights. Although this results in the possibility of not finding trust views to build a suggestion, it will at least provide some protection against malicious clients performing a Sybil attack.
As the client application only incrementally learns about the user, the trust view in the bootstrapping phase may not be indicative for the user's behaviour. The bound on which trust views are pruned must therefore be automatically adjusted. It will be increased during bootstrapping so that at the beginning - when the system does not yet know a lot about the user's browsing behaviour - a broad variety of assessments is accepted for recommendation. Then, while the bootstrapping proceeds, the recommendations become increasingly adapted to the user.
The adaption function for b was derived from a study conducted in [16], which analysed 26 browsing histories for 22 users. The adaptation described in [13] came to the following values for b

$$b = \begin{cases} 0.2223 * \ln(h) - 0.5036 & h < 352 \\ 0.8 & else \end{cases}$$

where h is the number of observed hosts so far [13].

Disclosure of client-information

The information gathered by the system is highly sensitive data, as it reveals the end entities with which the user has had a secure communication. This can be used to profile a person by their browsing habits. Although as long as the data is stored locally it is less security-critical, it does become an issue once the data is uploaded to the SP. Therefore the following decisions

have been taken:
The end certificates (trusted and untrusted) should be stored encrypted as they hold the most private information. The intermediate (CA) certificates do not pose such a threat, because CAs sign certificates for various services. An attacker therefore can not gain much information from this certificate [13].
In order to implement the recommendation system and the pushes (see section 3.3), the information needed in a single user's trust view are solely the trust assessments, a list of trusted and a list of untrusted certificates. Although the client does store a lot more information (the watch-list, revocation-information, ...etc.), this information is neither necessary for the system's functionality nor is it recommendable to be stored and updated on the SP regularly for two reasons. The first reason being that it can give information about the used end-entities thus being a privacy-issue, but more importantly the second reason being efficiency considerations. The information mentioned above is updated regularly by the local system itself, which could cause a massive increase on requests to the SP, making it difficult for it to answer all requests in reasonable time.

3.2.3 Outlook

This chapter holds functionalities, which have not yet been implemented, together with some considerations on the realisation.

Further security

To further secure the SP against sybil attacks, the SP could monitor anomalies in changes on trust views. For example, the number of trust view uploads within a certain timespan or large changes on single trust views, as they could be part of an attack. That this is possible is shown in [42].
Another aspect could be to only accept trust assessments and certificates of a certain age on the SP. Thus the timespan in which a MitM attack involving an (usually newly) issued fraudulent certificate is lengthened, which increases the likelihood for detection and revocation [13].

3.3 Pushes

It was already shown that many attacks on the reliability of CAs are not disclosed to the public. This results in the ongoing trust in this CA and possibly wrong future decisions. In order to prevent this the following chapter will propose a behavioural change detection scheme as well as a push service to warn other relying entities.

3.3.1 Functionality

In order to detect fraudulent certificates the system takes advantage of the fact, that many users with different trust views use the service. We assume that for any fraudulent certificate there is at least one user that does not trust this certificate by default and will therefore try to validate the certificate with his trust validation algorithm (addressed in section 2.6). When this fails, he reports the certificate to the SP, which will in turn try to validate the certificate. If the SP as

well comes to the conclusion that the certificate is invalid, it will inform all clients, that have an assessment for this key and entity in their trust view [13].

Reporting untrustworthy certificates to the Service Provider

Whenever an entity ε_1 evaluates a certification path as untrusted, it proceeds as follows. Be $p = (C_1, ..., C_n)$ the certification path and url the url from which p was obtained.

1. ε_1 establishes a TLS connection to $\boldsymbol{SP}_1$ and authenticates itself
2. ε_1 sends the tuple (p, url) to $\boldsymbol{SP}_1$
3. $\boldsymbol{SP}_1$ confirms the report to ε_1 with a success message

As longs as ε_1 does not receive the success message from the server, it queues the push locally and retries to send it [13].

Processing a push as Service Provider

To prevent fraudulent push reports the SP on receiving a push first verifies whether it concurs with the assessment "untrustworthy". The verification is done rather extensively because in this context we consider some seconds of delay less crucial than fraudulent reports. Latter would lead to the defective suspending of a CA, which for the client results in non-justified page loading delays [13].
Therefore when a SP receives a push message it determines its opinion on the push report as follows.
Let $\boldsymbol{VS} = (\boldsymbol{VS}_1, ..., \boldsymbol{VS}_j)$ be a list of validation services with outputs $R_i = \boldsymbol{VS}_i(C) \in \{\text{trusted}, \text{untrusted}, \text{unknown}\}$ for $1 \leq i \leq j$ on input of a certificate C.

1. $\boldsymbol{SP}_1$ performs standard path validation (including revocation checking) on path p
2. If path validation fails, it returns *invalid*
3. Else
   a) Query all validation services VS_i for $1 \leq i \leq j$ for C_n and set $R_i = \boldsymbol{VS}_i(C_n)$
   b) if there exists a $i \in \{1, ..., j\}$ with $R_i =$ untrusted, it returns *valid*
   c) else if there exists a $i \in \{1, ..., j\}$ with $R_i =$ trusted, it returns *invalid*
   d) else it returns *unknown*

If the report validation outputs *valid*, $\boldsymbol{SP}_1$ pushes a warning to its client and the other SPs. If the report is *invalid*, it is discarded. In case the validity of a report is *unknown*, it is queued and retrieved later. The latter case (*unknown*) is very uncommon due to the usage of many different types of validation services [13].

Pushing warnings to relying entities

If the report is considered justified, the report is pushed to all known SPs. Moreover for a valid report $(p = (C_1, ..., C_n), url)$ the SP fetches all users whose trust views contain an assessment for the key pk of a CA CA certified in C_{n-1} and pushes a CA warning $(\boldsymbol{pk}, \boldsymbol{CA}, C_n)$ to them [13].

Processing CA warnings

On receipt of a warning $(\boldsymbol{pk}, \boldsymbol{CA}, C_n)$ the relying entity suspends $\boldsymbol{CA}$ (the issuer of C_n and the subject of C_{n-1}). This is carried out by resetting the trust assessment values of $(\boldsymbol{pk}, \boldsymbol{CA})$ to the initialization values and adding one negative experience. The untrusted certificate C_n is added to the list of untrusted certificates.
The certificate that certified the connection between $\boldsymbol{pk}$ and $\boldsymbol{CA}$ (in the validation path this was certificate C_{n-1}) is not marked as untrusted. Thus the trustworthiness of $\boldsymbol{CA}$ can be learned anew and it can even become trustworthy again. This is important as fraudulent behaviour on a CA's side may not necessarily arise from malicious intent, but also from (temporary) error. However until the CA $\boldsymbol{CA}$ becomes trustworthy again, none of its certificates are considered such without reconfirmation. Certificates that were previously evaluated as trustworthy are kept as such, because the CA's current behaviour does not have retroactive effects [13].

3.3.2 Challenges

Through the implementation of this part of the system the following challenges had to be addressed.

Server-Push or Client-Pull

Several possibilities for the realization of the pushes were suggested. The first basic decision to be taken is whether the warnings should be realised by server-push or client-pull. Server-push describes a scheme where the server sends the warning to the client asynchronously, while a client-pull means that the client (usually periodically) pulls the warning from the server [43].
In order to realize a server-push the server requires an address on which the client can react. Usually the client is a common computer connecting to the internet over a router, meaning that it has no static or globally accessible IP address. Therefore a TLS-connection is only possible from client to server, but not vice versa. An address however that is globally accessible is the user's email address. The server could send the pushes as email and the client would periodically log into the user's email account and check for warnings.
The client-pull on the other hand would require the client to periodically request the server to send the warnings. While this obviously raises the number of requests to the server, it does not increase the attack surface posed by including new systems (like email). However, instead of periodically asking the server, there are more expedient ways to realise this. Whenever the SSL-Listener is triggered, it usually causes the extended validation to be performed. Instead we now first request the server to send all warnings. The operations associated with fetching the warnings (on server side) and processing them (on client side), as well as the sending and receiving of the messages are not computational expensive and therefore acceptable in terms of performance. After fetching the warnings, the trust validation is performed as usual, but the database is already updated with the newly untrusted certificate and reset trust assessment.
This realisation of the client-pull ensures that all warnings are present when they are needed (in trust validation), while maintaining the number of requests to the SP in a reasonable range. Especially does the client not bother the SP when there is no need (e.g. because the user's browsing is restricted to non-HTTPS addresses).

3.3.3 Outlook

This chapter holds the functionalities, which have not yet been implemented, together with some considerations on the realisation.

Push Validation

The push verification done by the SP was not yet implemented. The path validation can be performed using [44], while the validation services should work similarly to those on the client.

Re-executing Actions

Two moments have been described, where there is the possibility of not being able to execute the required action at the given moment. One is when a user detects a untrustworthy certificate, tries to push a report to the SP, but it is not contactable. The second is when the SP tries to validate the report, but the validation scheme returns 'unknown'. In both cases the action should be queued and retried later. This can e.g. be implemented by a background thread that periodically checks this queue and tries to re-execute the required action, until it can be removed from the queue.

3.4 Evaluation

To evaluate the system, we first extend the attacker model from section 2.4.2 with new attack vectors introduced by the inclusion of SPs.

3.4.1 Extended attacker model

Additional to the usual MitM-attacker capabilities (as described in 2.4.2) the attacker $\boldsymbol{A}$ is now able to act as a user on a SP of his choosing. Meaning that he can upload and alter his own trust view on this SP. Furthermore he may also be able to compromise other users registered at this or other SPs.
The SP's is presumed guarded, meaning that neither data nor behaviour can be changed by an attacker. Moreover we assume the communication between client and SP to be secure.
We distinguish three overall goals: Self-promoting a certain CA, so that it appears more trustworthy to other entities. This greatly increases the possibility for a successful MitM attack by a fraudulent certificate that is unjustifiably considered trustworthy. In contrast, slandering describes the act of making a CA look less trustworthy than it is. While this does not directly help $\boldsymbol{A}$, it may aim at disrupting the system's functionality or tarnishing another entities reputation. A third possibility is referred to as whitewashing. It describes the effort to make an entity that is already known to be untrustworthy re-appear under a new, clean identity [13].

3.4.2 Attacks on CA-TMS with SPs

The only points where the user ε_1 changes its behaviour based on information from $\boldsymbol{SP}$, is when it receives recommendations for an assessment's issuer trust or a warning resulting from a push

to the SP. We will focus first on possible attacks to the reputation system and afterwards on attacks to the push system.

Reputation System Although we assumed ***A*** is not able to alter data on the SP undetected, he can act as a user and inject malicious trust views undetected. By malicious trust views we mean trust views that contain unjustified opinions about certificates or trust assessments. The trust views on the SP will influence the suggestions given to a requesting entity. As we consider the SP's system and communication with the client to be secure, there is only one other threat, which is a DoS attack, that would leave the server to its local information or retry later. This can however not be considered a direct threat, as it does not inject new (possibly malicious) information into the local system and therefore will not change the behaviour in an unsuspecting way. The only threat therefore is the injection of malicious trust views.

One malicious trust view will most likely not make a noticeable difference regarding the suggestion. Therefore to visibly change the suggestion aggregated on the server, ***A*** has to control several trust views. This kind of attack resembles a sybil attack. It can be achieved by several attackers collaborating or a single attacker controlling multiple accounts. Registering multiple accounts on one's own has been complicated by adding a captcha to the registration process, requiring ***A*** to either break the captcha or manually fill all registration forms.

But even the control over multiple accounts does not pose a great threat due to the trust view selection mechanism described in section 3.2.2. For the recommendations the SP only considers similar trust view. That means the trust views go into account with a weigh that depicts how similar the current trust view is with the one of the requesting user. As both the SP's and the user's local system as well as the TLS connection is considered secure, ***A*** does not have access to the user's trust view and therefore can not easily produce a trust view that is considered similar. Moreover we rely on the differentness of the user base, meaning that if an attacker manages to create a trust view similar to user ε_i's, this does not necessarily mean, that it is also similar to another user ε_j's (where $i \neq j$). Such an attack can therefore not have a great impact on the entire user base [13].

Push System As seen on the analysis of the reputation system, ***A*** can again act as a user and try to influence other relying entities. Because we again consider both the user's and the SP's system to be secure as well as their communication channel, ***A*** is not able to impersonate the SP to the user. It is therefore not possible for ***A*** to push a warning directly to the user on behalf of the SP, as long as they communicate over the protected TLS-connection. Instead he can try to push fraudulent reports to the SP in hopes of them being evaluated as valid.

This however is unlikely due to the fact that the SP evaluates every push extensively. The assessment does not only include standard path validation, but also the querying of validation services. It is therefore highly improbable, that the push is evaluated as valid although it is fraudulent.

Synopsis It has been shown, that whitewashing is not a threat to the system, because new certificates are not automatically considered trustworthy. However, self-promoting and slandering can be aided by performing sybil attacks and injecting malicious trust views or finding schemes to push fraudulent reports that the SP somehow evaluates as correct. In both cases the SP could return erroneous information to the client, influencing the client in a malicious way. They can be obstructed by further complicating the registration process and monitoring suspicious behaviour [13].

In the following we give an overview over some simulations performed in [13], chapter 5.4.3 and 5.4.4, that prove the adequacy of the used methods. The simulations are based on 64 real browsing histories. In the following we show how they demonstrate the suitability of the Jacard Similarity Index for pre-selection and weighing of trust views for the computation of recommendations. Afterwards we address the detection of behavioural changes, as well as the improvement of information for the client, the influence on the attack surface and the impact on data loads and communication overhead.

Suitability of JSI It could be shown, that the JSI is suitable for pre-selection and similarity weighting.

To show the suitability for pre-selection the study examined the probability that a certain trust assessment (for a given CA and key) is contained in a trust view of the selected set. The result demonstrated that the probability grows with the similarity computed by the JSI. Therefore the probability for losing information in form of pruned trust views that contain the sought trust assessment is negligible.

On the aspect of the JSI's suitability for similarity weighting it was examined how much the selected trust views differ on the subject of the sough trust assessment. It could be shown that the higher the result of the JSI, the more similar the expectation of opinions on the common assessments. Therefore the usage of JSI as weights for the cFusion is appropriate [13].

Improvement of Information The simulations also showed how the extensions of CA-TMS with SPs significantly improved information-gathering and sped up the trust view's bootstrapping.

The first used measure is the reconfirmation rate. It is defined as follows

$$RRate = \frac{h_r}{h}$$

where h_r is the number of hosts where the certificate had to be reconfirmed and h is the total number of hosts. It depicts the need for external reconfirmation due to the lack of local information. Therefore the lower this number, the higher the amount of local information. It could be shown that for all possible security levels the reconfirmation rate with SPs lies significantly lower than the one without SPs. Especially at the beginning, where a very low amount of information is available locally.

A second measurement is the number of hosts that need to be observed until a comparable level of reconfirmation rate is reached. It could be shown that due to more rapid addition of local information when using a SP, the number of observed hosts until a certain reconfirmation rate level is reached, is significantly lower with the SPs than without [13].

Detection of behavioural changes The simulations were put in a scenario, where the attacker $\boldsymbol{A}$ had obtained a fraudulent certificate issued by a CA $\boldsymbol{CA}$. From the attacked user base q was specified as the percentage of users that considered $\boldsymbol{CA}$ trustworthy and was called trust rate. Then the simulations measured how many attacks $\boldsymbol{A}$ was able to perform until it was discovered. It was discovered that even on $q = 0.99$ (99% trust $\boldsymbol{CA}$) on average only 99 entities could be attacked successfully before detection. Additionally it showed that from the 306 different CAs found in the 64 trust views even with security level l_{min} the number of CAs that reached a trust rate $q \geq 0.1$ was only 26.8%. For l_{max} this value shrunk to 9.8%, while none of the CAs scored a $q = 0.9$. Additionally all CAs that are not part of the 306 superset of CAs are

immediately discovered. It is therefore highly unlikely, that ***A*** should be able to compromise a CA and go undetected for long [13].

Influence on the attack surface The simulations for the attack surface did not collect any negative experience and their findings therefore provide an upper bound.

It could be shown, that the querying of the SP speeds the information gathering up to their final state. This was shown by measuring the expectation values $E(o_{it}^{ca})$ and $E(o_{it}^{ee})$ and afterwards computing the deviation from the final state. The variation was at all times lower than 0.2 and was steadily decreasing after the first 10 to 30 observed hosts.

Moreover the number of trusted CAs was compared with and without the usage of SPs. This number can be considered the size of the attack surface. It could be shown the the number of trusted CAs and therefore the attack surface with SPs was slightly higher than without, but fades with the number of experiences collected. In comparison with the system-centric PKI (not the CA-TMS) system it could be shown that the additionally trusted CAs account for an increase of 1%, while the bootstrapping process could be sped up by 50%.

For the push service the probability of the attacker's success was compared for the different systems system-centric, user-centric (CA-TMS without SPs) and user-centric with SPs. As already established, q denotes the trust rate, meaning the percentage of users that trust the given CA. Moreover we define n as the number of entities, that were previously attacked by ***A***. It could be shown, that for the system-centric CA the success probability lies at 100%, while this decreases to q with the user-centric system and even q^n with CA-TMS with push-system [13].

Data load and communication overhead While the usage of the recommendation system decreases the need for external reconfirmation, it obviously increases the number of requests to the SP. The traffic overheads and data loads are compared for the reputation system. We assume the messages to the SP generally smaller or equal than the ones to a validation service. This is justified because latter require a certificate to be sent, while the SP only takes a pair (k, CA). The same applies to the response messages. For the simulation the number of reconfirmations was counted, while the number of requests to the recommendation system is already given by the number of CAs.

Although the findings showed that the absolute number of queries to the SP was higher than the one to the validation services, it is possible to let the recommendations run in the background, without blocking the page loading time. The reason for the higher number is that the recommendation is queried regardless of the trust validation's outcome.

Moreover the simulation found that a user on average only discovers seven new CAs per month. Therefore, assuming a user base of one million users this results in 162 requests per minute. This should be unproblematic, as the HTTPServlet is able to run multi-threaded and the database has been set to a maximum of 16 concurrent connections. [13].

4 Conclusion

This chapter holds the conclusion. It is started by a brief overview over the implemented system, followed by the open topics and an outlook.

4.1 Implementation

The existing CA-TMS system has been extended by the concept of SPs. This includes functionalities for user-management and uploading and regularly updating the mirrored trust view on the server. Furthermore the reputation system has been developed. Before a client adds a new trust assessment, he can query the SP, at which he is registered to receive an aggregated suggestion on the issuer trust. The trust views weighing into this suggestions are preselected to make the suggestion more relevant to the user. Thus the user is given more sensible information for the new trust assessment's initialization. Another system that has been developed is the push system. Once a client discovers a certificate he considers untrustworthy, he can report this to the SP. The SP then is supposed to validate this request, which has not yet been implemented. What has however, is that on verifying the report as valid, the SP finds all users that are affected by this certificate and saves the according warnings. Users regularly (always before trust validation) check for warnings and process them by resetting the crucial trust assessment and changing the discovered certificate to untrusted.

4.2 Evaluation

It could be shown that the addition of service providers while slightly increasing the attack surface by 1%, could also speed up the bootstrapping by 50% [13]. While sybil attacks can never be fully prevented, it was shown that the reputation system is vulnerable to them only by a small extent. At the same time the push system's detection mechanism proved to be highly efficient. Even if a fraudulent certificate is considered trustworthy by 90% of the 64 relying entities, the assault was detected after at average 9 successful attacks, while at the same time none of the considered CAs was trusted by 90% of the given group [13].

4.3 Open topics and future work

Some functionalities could not be put into practice. Those already described in [13] include the communication between different SPs and the push validation.
Other not directly described features could aim at further complicating and securing the registration process (e.g. by forcedly using email addresses as usernames) as well as monitoring suspicious behaviour.
Apart from features securing the already implemented systems (reputation and push system), another bigger system could be a synchronisation system. As the trust view aggregated on the user's computer is already customized to his needs, it makes sense reusing and synchronizing the trust view between different computers.

Abbreviations

AES	Advanced Encryption Standard
CA	Certification Authority
CA - TMS	Certification Authority - Trust Management System
CRL	Certification Revocation List
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IT	Information Technology
JSI	Jacard Similarity Index
JSON	Javascript Object Notation
MitM	Man in the middle
OCSP	Online Certificate Status Protocol
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
RA	Registration Authority
REST	Representation State Transfer
RSA	Rivest Shamir Adleman
SOA	Service Oriented Architecture
SP	Service Provider
SQL	Structured Query Language
SSL	Security Sockets Layer
TLS	Transport Layer Security
XML	Extensible Markup Language

Bibliography

[1] *Statista - Prognose zum Datenvolumen des privaten und geschäftlichen IP Traffics*. http://de.statista.com/statistik/daten/studie/266885/umfrage/prognose-zum-datenvolumen-des-privaten-und-geschaeftlichen-ip-traffics-weltweit/. [online; accessed 30-March-2016].

[2] *Statista - Global Internet Usage*. http://www.statista.com/study/12322/global-internet-usage-statista-dossier/. [slide 28; online; accessed 30-March-2016].

[3] C. P. Pfleeger, S. L. Pfleeger, and J. Margulies. *Security in Computing*. 5th ed. page 7. Prentice Hall, 2015.

[4] C. P. Pfleeger and S. L. Pfleeger. *Analyzing Computer Security*. 1st ed. pages 484–485 and 601–603. Prentice Hall, 2012.

[5] L. Wang and Kangasharju. J. "Real-world sybil attacks in BitTorrent mainline DHT". In: *Global Communications Conference (GLOBECOM), 2012 IEEE*. 2012, pp. 826–832. DOI: 10.1109/GLOCOM.2012.6503215.

[6] Ramez A. Elmasri and Shankrant B. Navathe. *Fundamentals of Database Systems*. Ed. by Carter Shanklin. 6th. pages 87–88. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2011.

[7] Diksha Gautam Kumar and Madhumita Chatterjee. "MAC based solution for SQL injection". In: *Journal of Computer Virology and Hacking Techniques* 11.1 (2014), pp. 1–7. DOI: 10.1007/s11416-014-0219-6.

[8] Kuo Zhao et al. "Improved Defense Against Domain Name Server Man-in-the-Middle Spoofing". In: *JOURNAL OF COMPUTATIONAL AND THEORETICAL NANOSCIENCE* 9.10 (2012), pp. 1750–1756. DOI: 10.1166/jctn.2012.2276.

[9] J. Buchmann. *Einführung in die Kryptographie*. 4th ed. pages 61, 133–134 and 203. Springer-Verlag Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-74452-8.

[10] A. Hülsing. "Practical Forward Secure Signatures using Minimal Security Assumptions". PhD thesis. Technische Universtität Darmstadt, 2013.

[11] J. Buchmann. *Einführung in die Kryptographie*. 5th ed. pages 139–161 and 201. Springer-Verlag Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-11186-0.

[12] J. Buchmann. *Introduction to Public Key Infrastructures*. 1st ed. pages 21–22, 75–76, 84 and 117–119. Springer Heidelberg New York Dordrecht London, 2013. DOI: 10.1007/978-3-642-40657-7__1.

[13] J. Braun. "Maintaining Security and Trust in Large Scale Public Key Infrastructures". pages 14, 32–35, 41–47, 70–76, 97 and 103–134. PhD thesis. Technische Universtität Darmstadt, 2015.

[14] *RFC 5246*. http://www.rfc-base.org/txt/rfc-5246.txt. [online; accessed 26-February-2016].

[15] C. Soghoian and S. Stamm. “Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL (Short Paper)”. In: *Proceedings of the 15th International Conference on Financial Cryptography and Data Security*. FC’11. Gros Islet, St. Lucia: Springer-Verlag, 2012, pp. 250–259. DOI: 10.1007/978-3-642-27576-0_20.

[16] J. Braun and G. Rynkowski. “The Potential of an Individualized Set of Trusted CAs: Defending against CA Failures in the Web PKI”. In: *Social Computing (SocialCom), 2013 International Conference on*. 2013, pp. 600–605. DOI: 10.1109/SocialCom.2013.90.

[17] C. Ellison and B. Schneier. “Ten Risks of PKI: What You’re not Being Told about Public Key Infrastructure”. In: *Computer Security Journal* 16.1 (2000), pp. 1–7.

[18] https://www.verisign.com/en_US/domain-names/com-domain-names/index.xhtml. [online; accessed 04-March-2016].

[19] R. Holz et al. “The SSL Landscape: A Thorough Analysis of the x.509 PKI Using Active and Passive Measurements”. In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC ’11. Berlin, Germany: ACM, 2011, pp. 427–444. DOI: 10.1145/2068816.2068856.

[20] Maryam Asadzadeh Kaljahi, Ali Payandeh, and Mohammad Bagher Ghaznavi-Ghoushchi. “TSSL: improving SSL/TLS protocol by trust model”. In: *SECURITY AND COMMUNICATION NETWORKS* 8.9 (2015), pp. 1659–1671. DOI: 10.1002/sec.1113.

[21] Roger Clarke and Xamax Consultancy. “The fundamental inadequacies of conventional public key infrastructure”. In: *Proceedings of the European Conference on Information Systems*. 2001, pp. 148–159.

[22] *Certificate Pinning Extension for HSTS*. https://www.ietf.org/mail-archive/web/websec/current/pdfnSTRd9kYcY.pdf. [online; accessed 2-March-2016].

[23] D. Gambetta. *Trust: Making and Breaking Cooperative Relations*. pages 213–234. B. Blackwell, Oxford, UK; Cambridge, Mass., USA, 1990.

[24] Audun Jøsang, Roslan Ismail, and Colin Boyd. “A Survey of Trust and Reputation Systems for Online Service Provision”. In: *Decis. Support Syst.* 43.2 (2007), pp. 618–644. DOI: 10.1016/j.dss.2005.05.019.

[25] D. H. Mcknight and N. L. Chervany. *The meanings of trust*. Tech. rep. University of Minnesota, 1996.

[26] S. Ries et al. “Trust and Trustworthy Computing: 4th International Conference, TRUST 2011, Pittsburgh, PA, USA, June 22-24, 2011. Proceedings”. In: ed. by Jonathan M. McCune et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. Chap. CertainLogic: A Logic for Modeling Trust and Uncertainty, pp. 254–261. DOI: 10.1007/978-3-642-21599-5_19.

[27] S. M. Habib et al. “Fusion of Opinions under Uncertainty and Conflict – Application to Trust Assessment for Cloud Marketplaces”. In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. 2012, pp. 109–118. DOI: 10.1109/TrustCom.2012.165.

[28] J. Braun et al. “Public Key Infrastructures, Services and Applications: 10th European Workshop, EuroPKI 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers”. In: ed. by Sokratis Katsikas and Isaac Agudo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. Chap. Trust Views for the Web PKI, pp. 134–151. DOI: 10.1007/978-3-642-53997-8_9.

[29] H. Cai, J. Vieten, and P. Weisenburger. *CA-TMS Dokumentation*. Technische Universtität Darmstadt. 2014.

[30] H. Asghari et al. “Security Economics in the HTTPS Value Chain”. In: *Twelfth Workshop on the Economics of Information Security (WEIS 2013)*. Washington, D.C., 2013.

[31] *CAcert Wiki. History of Risks & Threat Events to CAs and PKI*. http://wiki.cacert.org/Risk/History. [online; accessed 25-February-2016].

[32] Laura J. White et al. “Maintenance of service oriented architecture composite applications: static and dynamic support”. In: *JOURNAL OF SOFTWARE-EVOLUTION AND PROCESS* 25.1 (2013), pp. 97–109. DOI: 10.1002/smr.568.

[33] Mike P. Papazoglou and Willem-Jan van den Heuvel. “Service oriented architectures: approaches, technologies and research”. In: *VLDB JOURNAL* 16.3 (2007), pp. 389–415. DOI: 10.1007/s00778-007-0044-3.

[34] R.T. Fielding. “REST: Architectural styles and the design of network based software architectures”. pages 86–87. PhD thesis. University of California, Irvine, CA, USA, 2000.

[35] *RFC 7231*. https://tools.ietf.org/html/rfc7231. [section 8.3.1; online; accessed 26-February-2016].

[36] B. Canvel et al. “Advances in Cryptology - CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings”. In: ed. by Dan Boneh. Springer Berlin Heidelberg, 2003. Chap. Password Interception in a SSL/TLS Channel, pp. 583–599. DOI: 10.1007/978-3-540-45146-4_34.

[37] *RFC 2617*. http://www.rfc-editor.org/pdfrfc/rfc2617.txt.pdf. [online; accessed 26-February-2016].

[38] X. Wang and H. Yu. “How to Break MD5 and Other Hash Functions”. In: *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT’05. Aarhus, Denmark: Springer-Verlag, 2005, pp. 19–35. DOI: 10.1007/11426639_2.

[39] http://crackstation.net/hashing-security.htm. [online; accessed 2-Novembre-2015].

[40] https://developers.google.com/recaptcha/intro. [online; accessed 26-February-2016].

[41] https://evernotefolios.wordpress.com/2012/04/27/multiple-email-addresses-with-one-gmail-account/. [online; accessed 26-February-2016].

[42] Y. Yang et al. “Securing rating aggregation systems using statistical detectors and trust”. In: *IEEE Transactions on Information Forensics and Security* 4.4 (2009), pp. 883–898. DOI: 10.1109/TIFS.2009.2033741.

[43] E. Bozdag, A. Mesbah, and A. van Deursen. “A Comparison of Push and Pull Techniques for AJAX”. In: *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*. 2007, pp. 15–22. DOI: 10.1109/WSE.2007.4380239.

[44] http://docs.oracle.com/javase/8/docs/technotes/guides/security/certpath/CertPathProgGuide.html. [online; accessed 20-February-2016].

A Appendix

A.1 Statement

Thesis Statement pursuant to § 22 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. In the submitted thesis the written copies and the electronic version are identical in content.

Date: Signature: