CA trust management for the Web PKI [⋆]

Johannes Braun, Florian Volk, Jiska Classen, Johannes Buchmann and Max Mühlhäuser

Technische Universität Darmstadt/CASED
{jbraun|buchmann}@cdc.informatik.tu-darmstadt.de,
florian.volk@cased.de, jiska.classen@seemoo.tu-darmstadt.de, max@informatik.tu-darmstadt.de

Abstract. The steadily growing number of certification authorities (CAs) assigned to the Web Public Key Infrastructure (Web PKI) and trusted by current browsers imposes severe security issues. Apart from being impossible for relying entities to assess whom they actually trust, the current binary trust model implemented with the Web PKI makes each CA a single point of failure and creates an enormous attack surface. In this article, we present CA-TMS, a user-centric CA trust management system based on *trust views*. CA-TMS can be used by relying entities to individually reduce the attack surface. CA-TMS works by restricting the trust placed in CAs of the Web PKI to trusting in exactly those CAs actually required by a relying entity. This restriction is based on locally collected information and does not require the alteration of the existing Web PKI. CA-TMS is complemented by an optional *reputation system* that allows to utilize the knowledge of other entities while maintaining the minimal set of trusted CAs. Our evaluation of CA-TMS with real world data shows that an attack surface reduction by more than 95% is achievable.

1 Introduction

The Web Public Key Infrastructure (Web PKI) is one of the largest and most important cryptographic systems. The core of the Web PKI is the ecosystem of certification authorities (CAs) that are responsible for the issuance and the maintenance of digital certificates. These certificates are issued to web service providers and are used in the transport layer security (TLS) protocol [12] for authentication. Thus, the Web PKI enables authentication of web servers and subsequently, the establishment of secure connections between web browsers and services like e-banking or e-commerce, in which privacy, confidentiality, and integrity are often indispensable. For further details on the Web PKI and its security model, please refer to Section 2.1.

However, the Web PKI fails in many points to provide the desired security [16, 20, 21, 54]. One serious problem is that the security of the Web PKI does not scale with the enormous size of the Internet. For the sake of interoperability as much legitimate web service certificates as possible should be verifiable. Therefore, the number of CAs that are fully trusted by default in current browsers and operating systems has continuously been growing over the past. Currently, there are approximately 1590 trusted CAs controlled by 683 organizations and located in 57 different countries [13]. As each of these trusted CAs can sign certificates for any web service or domain, trusting a single malicious CA that is, in fact not trustworthy, can break the whole Web PKI's security. An attacker who is in possession of a forged certificate that was signed with the key of one of the trusted CAs can potentially intercept the complete communication between any Internet user and the certified web server without the user even noticing the attack. Thus, with each additional CA, the risk of trusting a

[⋆] Manuscript version. The final version of this paper appears in the Journal of Computer Security, IOS Press malicious or defective CA increases because the attack surface of the Web PKI grows. The history of security incidents clearly demonstrates that this is more than just a hypothetical threat [8, 40, 9, 23, 18, 22, 20, 14, 48].

Blacklisting CAs or revoking malicious certificates are the reactions to security incidents. However, as these mechanisms are reactive, they have an inherent delay exposing the entities at risk until the detection of the threat. As explained, the attack surface grows proportional to the number of CAs an entity trusts. However, recent experiments with browser histories of different human users have shown that on average, a user only requires a small subset of CAs to be able to validate the certificates of all web services he uses. The size of this subset lies in the range of 10% of the CAs available and trusted by default in the Web PKI [3].

In this paper, we describe a new approach to reduce the attack surface by reducing the number of trusted CAs to those that are really required by the entities. Additionally, among the trusted CAs, we introduce variable trust levels to enable more fine grained trust decisions. According to the value-at-stake, a trust level is evaluated as sufficient or insufficient to consider a connection to a web service as secure.

To achieve this, we present a *CA trust management system (CA-TMS)*. The core of CA-TMS are *trust views* (first presented in [4]) that serve as a local and user dependent knowledge base for trust decisions. This is complemented by an (optional) reputation system for the trustworthiness of CAs. We illustrate the mechanisms for the establishment and the management of the trust view. Moreover, we implement learning processes and define decision rules by employing computational trust models. The real trustworthiness of CAs is approximated by subjective probabilistic trust values. CA-TMS allows an adaptation of the number of trusted CAs to the requirements of the relying entity and automated trust decisions based on defined decision rules. In our solution, we focus on applicability, thus we only use data which is already available or will be collected over time. However, the system is open for extensions with additional information sources. We build on the techniques of previous works like public key pinning and certificate notaries and combine different mechanisms as building blocks to solve separate sub problems. Different from those existing mechanisms, CAs in the entity's trust view have different trust levels and might even be fully trusted depending on the context. Furthermore, trust evaluation is based on local experiences of the entity, not requiring recommended trust values embedded in certificates or the evaluation of certificate policies and expert opinions. Thus, our solution can work autonomously and does not require an additional check of every (new) certificate. CA-TMS provides a trade-off between overhead due to reconfirmations and solely relying on CAs. Furthermore, the management of local experiences guarantees that CAs are only trusted after they have previously been encountered and checked. A CA is only trusted when the entity needs this CA to contact a web service—independently from the CA's global reputation. This protects the entity from malfunctions of CAs that in general follow good security practices but are actually irrelevant for the entity itself.

As said, trust views require the collection of experiences. I order to speed up and improve the initial collection process, we extend CA-TMS by a reputation system. This is designed under the prerequisite to provide recommendations tailored to the specific entity's requirements in order not to weaken the principle to trust in a individually minimized set of CAs.

The paper is organized as follows. Section 2 describes the Web PKI and our security model. Background on the computational trust model CertainTrust and further related work is given in Section 3. Afterward, in Section 4, we describe trust views and their mechanisms in detail. The subsequent Section 5 introduces a reputation system that completes CA-TMS. In Section 6, we evaluate CA-TMS in respect to performance and security and discuss limitations. We end with conclusions and future work in Section 7.

2 Web PKI & security model

2.1 The Web PKI

Secure Internet connections between web browsers and web servers in general rely on public key cryptography to authenticate web servers and establish session keys. Public key cryptography requires key pairs: a private key that is only known to the owner of the key pair (in our case a web server) as well as a public key, which must be known and authenticated to everyone who wants to establish a secure connection to the owner of the associated private key. A public key is bound to an identity via a digital *certificate* according to the X.509 standard [10]. Whenever a relying entity contacts a web server and successfully establishes a TLS connection using the public key in the web server's certificate, the relying entity can be sure that the web server knows the private key matching the public one. As the certificate binds the public key to an identity, the relying entity can be sure about the authenticity of the web server.

As it is impossible to exchange certificates directly between all web servers and all browser users (the relying entities), the Web PKI uses a hierarchical but tightly interwoven structure of CAs that digitally sign certificates. If a certificate is signed by a trusted CA, the authenticity of a web server that employs the certificate is transitively trusted. The Web PKI has a set of *Root CAs*. Their public keys are usually distributed within trusted lists called *root stores*, along with operating systems and browsers. The Root CAs act as basis for the whole PKI. Root CAs sign certificates for *subordinate CAs* (Sub CAs) which themselves sign certificates for other Sub CAs and end entities such as web servers, which we also refer to as hosts. This way, a hierarchical structure is created. The chain of certificates starting with a Root CA's certificate (usually self-signed) and ending with a web server's certificate is called *certification path*. The process of checking the certification path for correctness and validity is called *path validation* [10]. *Self-signed certificates* are certificates that are issued by an entity to itself and are signed with the private key associated to the public key in the certificate. Self-signed certificates may evolve as roots of certification paths (authenticated via root stores) or as intermediary certificates within a path (mainly for compatibility reasons). A relying entity obtains the certification path from a web server during the establishment of a TLS connection.

For a relying entity, in order to be convinced of the *key legitimacy* of a public key k, namely to be convinced whether a public key k in a certificate belongs to the identity contained in the subject field of the certificate, two things are required [34, 45, 63]. First, the relying entity must be convinced of the key legitimacy of the CA's key that was used to sign the certificate that contains k. Second, the relying entity must trust the CA to issue trustworthy certificates. The latter is called *issuer trust* in the CA.

In the Web PKI, issuer trust and key legitimacy are binary. Any certificate signature that can be verified using the root store is absolutely trusted. This is because in the current trust model, by signing a Sub CA's certificate, the superordinate CA delegates all its privileges to the Sub CA, which leads to the above mentioned 1590 CAs fully trusted to sign certificates for any web server. Although CAs achieve different qualities of service, this is currently not reflected within trust decisions. Different CAs implement different schemes to verify identities of the key owners they sign certificates for and employ different security mechanism. But, for example, a certificate containing a superficially verified identity appears to be as trustworthy as a certificate where the contained identity was checked thoroughly.

2.2 Security model

In the model exist two entities e_1 and e_2. e_1 establishes a TLS connection to e_2. The problem is to decide if the connection is trustworthy for e_1.

A connection is trustworthy for e_1 if the public key k of e_2 that was used in the TLS connection establishment is trusted by e_1 to be a valid public key of e_2. This requires:

1. e_1 has a valid certificate C that binds k to e_2.
2. e_1 trusts in the issuer of C.

The first requirement is a standard PKI issue. To fulfill requirement 1, e_1 needs to have a certification path $p = (C_1, ..., C_n)$ such that:

(a) $C_n = C$
(b) p passes path validation

Requirement 2 is fulfilled if p additionally passes *trust validation*. Explicit trust validation is not incorporated in the current deployment of the Web PKI. We show how this can be realized with *trust views* and the proposed CA trust management system and explain how this allows to reduce the number of actually trusted CAs and therewith the risk of relying on maliciously issued certificates. We first introduce computational trust and present related work.

3 Computational trust & related work

3.1 Background on computational trust

Computational trust is a means to support entities in making decisions under uncertainty, e.g., under incomplete information. The two most widely-used definitions of trust are *reliability trust* and *decision trust*. Gambetta describes *reliability trust* as a subjective probability of performance without the option to monitor the performed action [19]. While this definition only captures the beliefs of an entity that is potentially unaffected by its trust, Jøsang defines *decision trust* as the will to depend on a trusted entity (Jøsang in [38], inspired by McKnight and Chervany [47]):

> *Decision trust is the extent to which a given party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.*

Starting from recommendations, experiences from previous interactions, and context-related indicators of trustworthiness, computational trust models calculate an approximation for the quality of future interactions. For this paper, the CertainTrust trust model by Ries [52] is used. CertainTrust was extended with CertainLogic, a set of operators to combine CertainTrust opinions. These operators are similar to those of propositional logic, but consider the inherent uncertainty of CertainTrust opinions.

CertainTrust can handle two ways of expressing trust-related information:

– The *experience space* collects results from interactions as binary experiences, i.e., an interaction was either positive or negative.

– The *opinion space* uses a triple (t, c, f) to express an opinion o_S about a statement S. The value $t \in [0; 1]$ represents the trust in the correctness of the statement, while the certainty $c \in [0; 1]$ represents the probability that t is a correct approximation. c scales with the amount of information (for example, the number of collected experiences): the more information available, the more reliable is the approximation. Finally, $f \in [0; 1]$ defines a context-specific, initial trust value in case no information was collected, yet. This parameter serves as a baseline and represents *systemic trust*. Besides that, CertainTrust has a system-wide parameter n, which defines how many experiences are expected on average for a statement. This means after the collection of n experiences, for one opinion, the opinion's certainty is 1. We set $n = 10$. The effects of the parameter choice for n are discussed in Section 6.1.

There exists an ambilateral mapping between the experience space and the opinion space by parameterizing a Bayesian probability density function with the amount of positive and negative experiences. For details, see [51]. In this paper, trust information is collected in the experience space while the opinion space is used to combine trust statements about different CAs.

A CertainTrust opinion represents a statistical estimate that the next experience will be a positive one. Based on the experiences already collected, a maximum likelihood estimate for the next experience is given by the trust value. The concordant certainty value represents the confidence in the estimate. Positive and negative experiences are weighted equally in order to supply an unbiased estimate for the binary experiences—thereby forming a binomial sample.

From opinions, an expectation can be computed. It represents the expectation for future behavior. In CertainTrust, the expectation of an opinion o_A is defined as

$$E(o_A) = t_A \cdot c_A + f_A(1 - c_A)$$

Herein, with increasing certainty (which means that a larger amount of experiences is available), the influence of the initial trust f ceases.

There are several operators to combine different opinions. From two opinions about two independent statements a combined opinion about the statement regarding the truth of both input statements is computed with the AND-Operator of CertainLogic [52]:

Definition 1 (CertainLogic AND-Operator). *Let A and B be independent statements and the opinions about these statements be given as $o_A = (t_A, c_A, f_A)$ and $o_B = (t_B, c_B, f_B)$. Then, the combined opinion on the statement regarding both A and B is defined as follows:*

$$o_A \wedge o_B = (t_A \wedge t_B, c_A \wedge c_B, f_A \wedge f_B) \text{ with}$$

$$\begin{aligned} c_{A \wedge B} &= c_A + c_B - c_A c_B \\ &\quad - \frac{(1 - c_A)\, c_B\, (1 - f_A)\, t_B + c_A\, (1 - c_B)\, (1 - f_B)\, t_A}{1 - f_A f_B} \\ \text{if } c_{A \wedge B} = 0\text{: } t_{A \wedge B} &= 0.5 \\ \text{if } c_{A \wedge B} \neq 0\text{: } t_{A \wedge B} &= \frac{1}{c_{A \wedge B}} (c_A c_B t_A t_B \\ &\quad + \frac{c_A(1 - c_B)(1 - f_A) f_B t_A + (1 - c_A) c_B f_A (1 - f_B) t_B}{1 - f_A f_B}) \\ f_{A \wedge B} &= f_A f_B \end{aligned}$$

The CertainLogic AND-Operator is commutative.

The AND-Operator can be used to combine independent opinions on statements. Besides combining independent opinions, there might be the need to combine dependent opinions, i.e., opinions on the same statement made by different entities. CertainLogic provides three FUSION-Operators for this task. We will only use the cFUSION-Operator [24], which combines opinions by taking their inherent conflict into account. For example, asking different entities about the trustworthiness of a CA A might result in two completely different opinions based on different experiences made. One opinion o_{A_1} might be positive with high certainty $c_{A_1} \approx 1$, while the other opinion o_{A_2} might be negative, also with high certainty $c_{A_2} \approx 1$. Obviously, these two opinions carry some conflict as they cannot both be correct at the same time. The cFUSION-Operator handles this conflict by lowering the certainty of the combination result. Other FUSION-Operators, e.g., [36, 24], do not account for conflict and only average the trust and certainty values of the resulting opinion. Furthermore, cFUSION allows to assign weights to input opinions to give them higher or lower importance. cFUSION is defined in [24]:

Definition 2 (CertainLogic cFUSION-Operator). *Let A be a statement and let $o_{A_1} = (t_{A_1}, c_{A_1}, f_{A_1})$, $o_{A_2} = (t_{A_2}, c_{A_2}, f_{A_2})$, $\ldots$, $o_{A_n} = (t_{A_n}, c_{A_n}, f_{A_n})$ be n opinions associated to A. Furthermore, the weights w_1, w_2, $\ldots$, w_n (with $w_1, w_2, \ldots, w_n \in \mathbb{R}_0^+$ and $w_1 + w_2 + \cdots + w_n \neq 0$) are assigned to the opinions o_{A_1}, o_{A_2}, $\ldots$, o_{A_n}, respectively. The conflict-aware fusion of o_{A_1}, o_{A_2}, $\ldots$, o_{A_n} with degree of conflict DoC is denoted as:*

$$o_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)} = ((c_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)}, t_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)}, f_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)}), DoC) \text{ with}$$

$$\text{if all } c_{A_i} = 1\text{: } t_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)} = \frac{\sum_{i=1}^{n} w_i t_{A_i}}{\sum_{i=1}^{n} w_i}$$

$$\text{if all } c_{A_i} = 0\text{: } t_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)} = 0.5$$

$$\text{if } \{c_{A_i}, c_{A_j}\} \neq 1\text{: } t_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)} = \frac{\sum_{i=1}^{n} (c_{A_i} t_{A_i} w_i \prod_{j=1,\, j\neq i}^{n} (1 - c_{A_j}))}{\sum_{i=1}^{n} (c_{A_i} w_i \prod_{j=1,\, j\neq i}^{n} (1 - c_{A_j}))}$$

$$\text{if all } c_{A_i} = 1\text{: } c_{\hat{\oplus}_c(A_1,A_2,\ldots,A_n)} = 1 - DoC$$

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

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

$$DoC = \frac{\sum\limits_{i=1,\ j=i}^{n} DoC_{A_i,A_j}}{\frac{n(n-1)}{2}}$$

$$DoC_{A_i,A_j} = \left|t_{A_i} - t_{A_j}\right| \cdot c_{A_i} \cdot c_{A_j} \cdot \left(1 - \left|\frac{w_i - w_j}{w_i + w_j}\right|\right)$$

The CertainLogic cFUSION-Operator is commutative.

3.2 Further related work

The multitude of problems and disadvantages of the currently deployed Web PKI is described by well known researchers [16, 20, 21]. Monitoring of the Web PKI reveals its enormous size and shows that indeed malpractices are common [13, 29, 15].

Many attempts exist to circumvent the problems imposed by possible CA failures and thus to enhance Internet security. Certificate or public key pinning (e.g., [17, 50]) means that relying entities store certificates of formerly accessed websites. Based on the *trust on first use approach*, it implies that a possible attacker must be present during the first connection establishment to a website and thus provides a clear security benefit by reducing the freedom of action for possible attackers. Unfortunately, this either implies that each CA is trusted equally in case a new web page is accessed or that the trust decision is transferred to the relying entity requiring it to have PKI expertise. Also, the approach suffers from the problem how and when to allow pinned certificates to be exchanged (e.g., due to certificate expiry), which again provides some room for an attacker.

Notarial solutions [31, 43, 6] maintain databases containing formerly observed certificates and can be queried to reconfirm the authenticity of a specific certificate. Some of the solutions also involve consensus decisions of several independent notary servers or employ multi path probing. When incorporating notaries to evaluate the quality of certificates, trust is deferred from CAs to the notaries or a majority of notarial services. While notaries are a powerful addition to the security of the Web PKI, they suffer from the problem of false positives, e.g., when hosts serve different certificates [39] like in the case of Google and Facebook or if new certificates are deployed, which have previously not been observed and are not in the database of the notaries. Furthermore, querying a notarial service for the validity of a certificate reveals the service an entity is trying to connect to and thus, reveals privacy relevant information. In contrast to local evaluation only, contacting notarial services during the setup of every secure connection introduces communication overhead delays that are undesirable from the entity's perspective.

Certificate transparency [41] is an experimental proposal for publicly logging the existence of TLS certificates to allow public auditing of CAs and the detection of erroneously issued certificates. Yet, it requires web page operators to monitor public logs and CAs to publish all issued certificates in order to function.

An often discussed alternative to the Web PKI is the binding of certificates – above all self-signed certificates – to Domain Name System (DNS) names using DNSSEC [28]. This mechanism removes many drawbacks of the current PKI system, yet security then relies on the security of the hierarchical DNS infrastructure which recreates many of the problems of the Web PKI. Furthermore, DNSSEC is still not completely accepted and requires major protocol changes and implementation efforts.

Recently, Kasten et al. [39] proposed CAge, which works by restricting the set of domains, for which a certain CA is trusted to sign certificates. Basically, they propose that browsers should realize the functionality that was originally intended with the name constraints extension [10] but is barely employed by CAs themselves. Based on an Internet-wide survey, Kasten et al. show that CAs generally sign certificates for a certain set of domains and that their behavior is stable in most cases which allows the restriction based on past behavior. It turns out that such a restriction could reduce the attack surface by 65%-90%. CAge is complementary to our approach. While CAge globally limits the power of each CA and therewith the impact of its failure, our approach limits the number of trusted CAs in a user-specific way.

The enhancement of PKI with trust computation has been proposed by many researchers. The CertainTrust model and CertainLogic [52] used by us are equivalent to the Beta Reputation System and Subjective Logic, both by Jøsang et al. [37, 35, 36], as these models both rely on binary experiences that are combined using a Bayesian approach with beta probability density functions. A survey on different trust models that rely on this computational approach and similar ones can be found in the surveys by Jøsang et al. [38] and Ruohomaa et al. [53]. Jøsang proposes an algebra for trust assessment in certification chains in [34] but mainly addresses trust networks similar to PGP [63]. Huang and Nicol [30] also define another trust model for trust assessment in PKI. Both approaches require trust values recommended by the intermediates to evaluate trust chains. Such recommendations are in general not included within commercial certificates. Different certificate classes like domain validated (DV) or extended validation (EV) can be indicators for such trust values, but are to coarse grained for trust evaluation and are also not CA specific.

Other researchers base trust evaluation in CAs on their policies and the adherence to those [7, 57]. This requires policy formalization [59, 7, 57] for automated processing. Such formalized policies are not provided by CAs, and are in general far to complex to be evaluated by the relying entities. Therefore, such approaches require technical and legal experts to process policies [58].

In general, by including recommendations, systems using computational trust can be extended toward trust management systems. A survey on systems that evolve local trust into global trust can be found in [2] and, more specific to reputation-based trust management systems, in [53, 42, 27].

In the following section, we present the concept of trust views and trust validation. It uses the CertainTrust model and while it employs certificate pinning and notarial solutions, also additional mechanisms like certificate transparency or DNSSEC can be integrated, e.g., as additional validation services used to reconfirm certificates. An optional reputation system for CA trust management is then explained in Section 5.

4 Trust view and trust validation

The purpose of establishing a trust view is to enable explicit trust validation thereby locally reducing the number of trusted CAs on a per-entity level. The differences in the trust needed for different applications are considered during trust validation. For example, there is a difference in the trust needed to visit a search engine and the trust needed to supply an online-shopping web site with your credit card information.

4.1 Challenges

The set of CAs required by an entity is not fixed but changes over time. The challenge herein is to establish and manage a trust view in a dynamic way. We identified the following constraints for dynamically updating the set of trusted CAs as well as for assigning trust levels to them:

1. **Minimal user involvement**: an informed assessment of the quality of a CA's certification processes is beyond the capabilities of the average Internet user [26, 55].
2. **Incomplete information on CA processes**: data on the quality of a CA's certification process might be incomprehensible, incomplete, or not available at all.
3. **Incomplete information on the entity's requirements**: in general, the web services that an entity will contact in the future are unknown and therefore also the CAs that are required to verify the certificates of such web services.

4.2 The trust view

For trust validation, entity e_1 has a *trust view* View. The trust view is the local knowledge base of e_1 and contains all previously collected information about other entities and their keys. It is built incrementally during its use for trust validation. The trust view of e_1 consists of:

- a set of trusted certificates
- a set of untrusted certificates
- a set of public key trust assessments

The trusted certificates are all certificates that have previously been used to establish a trustworthy connection to another entity. The untrusted certificates are those certificates, for which the connection was evaluated untrustworthy. Furthermore, there is one public key trust assessments for each pair of $(public\ key,\ CA\ name)$ that was contained in a previously evaluated certification path. A trust assessment represents all information collected for the respective pair during prior trust validations.

A public key trust assessment TA is a tuple $(k, ca, S, o_{kl}, o_{it}^{ca}, o_{it}^{ee})$, where

- k is a public key.
- ca is the name of a certification authority.
- S is a set of certificates for k. The subject of these certificates is ca. This set contains all the certificates with subject ca and public key k that have previously been verified by e_1.
- o_{kl} is an opinion. It represents the opinion of e_1 whether k belongs to ca or not (key legitimacy of k).
- o_{it}^{ca} is an opinion. It represents the trust of e_1 in ca to issue trustworthy certificates for CAs (issuer trust in ca when using k to sign CA certificates).
- o_{it}^{ee} is an opinion. It represents the trust of e_1 in ca to issue trustworthy certificates for end entities (issuer trust in ca when using k to sign end entity certificates).

In order to decide whether the connection to entity e_2 is trustworthy, entity e_1 runs the trust validation algorithm (cf. Section 4.4).

4.3 Initialization of trust assessments

A trust assessment $TA = (k, ca, S, o_{kl}, o_{it}^{ca}, o_{it}^{ee})$ is initialized whenever a pair $(public\ key, CA\ name)$, for which there is no trust assessment in the trust view View, is observed within a CA certificate C. We assume that a root store is available during initialization. Then, TA is initialized as follows:

- $k = public\ key$
- $ca = CA\ name$
- $S = \{C\}$
- $o_{kl} = (1, 1, 1)$ if the CA is a Root CA, else $o_{kl} = unknown$.
- The initialization of $o_{it}^{x}, x \in \{ca, ee\}$ is the following:
  1. If there exists $\tilde{TA} \in$ View such that $\tilde{ca} = ca$ and (ca is a Root CA or the issuing CA of C equals the issuing CA of one $\tilde{C} \in \tilde{S}$), then set $o_{it}^{ca} = \tilde{o}_{it}^{ca}$ and $o_{it}^{ee} = \tilde{o}_{it}^{ee}$.
  2. If there exists no such $\tilde{TA}$ then:
     (a) If for $1 \leq i \leq n$ there are trust assessments $TA_i \in$ View with $C_i \in S_i$, where the issuing CA of C_i is equal to the issuing CA of C, then compute $f^x = \frac{1}{n} \sum_{i=1}^{n} E(o_{it,i}^{x})$ and set $o_{it}^{x} = (0.5, 0, \min\{maxF, f^x\})$ for $x \in \{ca, ee\}$ and $maxF = 0.8$.
     min$\{a, b\}$ denotes the minimum of the input values.
     (b) Else set $o_{it}^{ca} = o_{it}^{ee} = (0.5, 0, 0.5)$.

The key legitimacy is set to complete ($o_{kl} = (1, 1, 1)$) for Root CA keys as these keys are confirmed via the root store. For other CA keys, key legitimacy is computed during trust validation as long as key legitimacy is $unknown$. During the evolution of the trust view, key legitimacy may be fixed and set to complete as soon as enough evidence has been collected. We discuss this in Section 4.5.

Step 1 of the o_{it}^{x} initialization realizes the transfer of earlier collected information about a CA to TA, which is especially relevant for CA key changes. Herein, the requirement of either being a Root CA, i.e., being authenticated via the root store, or having the same issuing CA ensures that the collected information undoubtedly belongs to the CA in question.

Step 2 provides an initialization mechanism if no prior information about the CA is available. If the new CA's key is certified by a CA that certified keys of several other CAs, i.e., there are siblings for which experiences have already been collected, we use the average over the expectations of the respective issuer trusts for initialization. The reason is that a CA evaluates a Sub CA before signing its key, and thus, these Sub CAs are assumed to achieve a similar level of issuer trust, like a stereotype [5]. While Burnett et al. apply machine learning techniques to identify the features that describe stereotypes, the solution presented here assumes that being certified by the same CA is the only relevant feature for stereotyping. Therefore, all known CAs that are certified by the same CA form a stereotype. We bound the initial trust value f by $maxF$ in order not to overestimate a CA's trustworthiness (cf. Section 6.1 for a discussion of the effects of the parameter choice for $maxF$). If also no siblings are available in the trust view, the issuer trust $o_{it}^{x} = (0.5, 0, 0.5)$ reflects that no experiences have been collected and that the CA may either be trustworthy or not.

Optimally, further information is collected for initialization. Our system is open for such extensions. In Section 5, we describe how to realize a reputation system, which recommends the issuer trust of a CA to an entity based on the trust views of other entities. Further information could be gathered from policy evaluation as, e.g., proposed by Wazan et al. [57, 58]. A drawback of this approach is its need for some kind of expert or expert system to evaluate the certificate policies and practice statements, because these documents cannot be processed automatically at the time being. So far, no such services are available in practice. Yet, given such additional data, it can be mapped into an opinion and integrated into the initialization process.

Bootstrapping Despite the fact that an entity will often access the same services and see the same CAs repeatedly [3], it takes a certain time until enough experiences are collected such that the system may operate autonomously (cf. Section 6.1 for details on the evolution of exemplary trust views). Therefore, a bootstrapping procedure is required to face possible delays and usability problems due to the involvement of additional validation services (cf. Section 4.4). A possibility for such a bootstrapping procedure is to scan the browsing history. From the history, the services that are accessed via a TLS connection can be identified and the respective certification paths can be downloaded. The paths can then be used to bootstrap the trust view. This initial bootstrapping is only to be performed once and afterward, the system can mainly fall back on the collected experiences.

4.4 Trust validation

We now describe the trust validation algorithm. It takes the trust view of entity e_1 and a certification path for the certificate of entity e_2 as input and computes the key legitimacy of e_2's key to decide whether a connection established with e_2's key is to be considered trustworthy. The decision depends on the security criticality of the application that is to be secured by the connection from e_1 to e_2. The information available in the trust view may not be sufficient to complete the trust validation. In such a case, validation services are used as a fall back mechanism. We present the detailed algorithm in the following:

Input:

- The certification path $p = (C_1, ..., C_n)$ without intermediary self signed certificates
- The trust view View of e_1
- A security level $l \in [0; 1]$ for C_n. l is selected by e_1 and represents the security criticality of the application that is to be secured by the connection from e_1 to e_2. The higher l, the more security critical is the application.
- A list of validation services $VS = (vs_1, ..., vs_j)$ with outputs $R_i = vs_i(C) \in \{trusted, untrusted, unknown\}, 1 \leq i \leq j$ on input of a certificate C.

Output: $R \in \{trusted, untrusted, unknown\}$

The algorithm proceeds as follows:

1. If C_n is a trusted certificate in View then $R \leftarrow trusted$
2. If p contains a certificate that is an untrusted certificate in View then $R \leftarrow untrusted$
3. If C_n is not a certificate in View then
   (a) For $1 \leq i \leq n-1$ set k_i to the public key in C_i and ca_i to the subject in C_i.
   (b) Initialize the trust assessments for pairs (k_i, ca_i) for which there is no trust assessment in View (as described in Section 4.3). Store the new trust assessments in the temporary list TL.
   (c) For $1 \leq i \leq n-2$ set $o_{kl,i}$ to the key legitimacy of k_i and $o^{ca}_{it,i}$ to the issuer trust (for CA certificates) assigned to k_i in View.
   (d) Set $o_{kl,n-1}$ to the key legitimacy of k_{n-1} and $o^{ee}_{it,n-1}$ to the issuer trust (for end entity certificates) assigned to k_{n-1} in View.
   (e) Set $h = \{max(i) : o_{kl,i} = (1, 1, 1)\}$

```
    (f) Compute $o_{kl,n} = (t, c, f) = o^{ca}_{it,h} \wedge o^{ca}_{it,h+1} \wedge \cdots \wedge o^{ca}_{it,n-2} \wedge o^{ee}_{it,n-1}$
    (g) Compute the expectation $exp = E(o_{kl,n})$
    (h) If $exp \geq l$ then $R \leftarrow trusted$
    (i) If $exp < l$ and $c = 1$ then $R \leftarrow untrusted$
    (j) If $exp < l$ and $c < 1$ then
         i. For $1 \leq i \leq j$ query validation service $vs_i$ for $C_n$ and set $R_i = vs_i(C_n)$.
        ii. Set $R_c$ to the consensus on $(R_1, ..., R_j)$, then $R \leftarrow R_c$.
    (k) Update View. (See Section 4.5 for details.)
4. Return $R$
```

According to previous works [34, 45, 63], the key legitimacy of a key is computed as the key legitimacy of the CA's key in conjunction with the issuer trust in the CA $issuer$: $o_{kl} = o_{kl,issuer} \wedge o_{it,issuer}$. The computation of the key legitimacy based on a certification path with length greater than one follows directly from chaining this rule and the fact that the key legitimacy of the first key k_h in the path is $o_{kl,h} = (1, 1, 1)$. Such a key always exists as this holds at least for Root CA keys. Thus, $o_{kl,n} = (t, c, f) = o^{ca}_{it,h} \wedge o^{ca}_{it,h-1} \wedge \cdots \wedge o^{ee}_{it,n-1}$ as for the CertainLogic AND operator holds if $o_A = (1, 1, 1)$ then $o_A \wedge o_B = o_B$.

Security levels Entity e_1 assigns security levels to classes of applications according to their value-at-stake (cf. [57] for a similar approach). That means, e_1 defines which security level the trust evaluation must achieve for the certification path in question in order to be accepted without further reconfirmation. Note, that e_1 does not rate the trustworthiness of applications.

A security level is a real number between 0 and 1. The higher the security level l is, the higher is the required key legitimacy for a connection to be evaluated trustworthy. The assignment of security levels is a subjective process and depends on the risk profile of e_1, which is out of scope of this paper. We propose to apply three classes of security levels $l_{max} = 0.95$, $l_{med} = 0.8$ and $l_{min} = 0.6$ (cf. Section 6.1 for details on the choice of security levels and the associated effects). An exemplary assignment of applications to the security levels could then be l_{max} for online banking, l_{med} for e-government applications, and $l_{min} = 0.6$ for social networks. Please refer to Section 4.4 for details how this input may be provided to CA-TMS for trust validation.

Validation services A certification path containing previously unknown CAs results in a low key legitimacy for the key certified in the end entity's certificate. On the one hand this is intended, as it leads to firstly distrusting in keys certified by unknown CAs. However, this is not necessarily due to malicious behavior, but due to a lack of information. Thus, whenever the key legitimacy is too low to consider a connection trustworthy, and the certainty is less than one, validation services like notary servers (cf. Section 3.2) are queried to reconfirm a certificate. Note that also solutions like certificate transparency or DNSSEC could be integrated here. If a certificate is reconfirmed to be authentic, the connection is considered trustworthy. If the validation services reply with $unknown$, i.e., it is unclear if the certificate is trustworthy or not, the algorithm outputs $unknown$. Only in this case, the relying entity is asked for a decision.

User interaction As stated in Section 4.1, CA-TMS aims at minimal user involvement. While there exists a user interface for CA-TMS, where experienced users may change the system parameters described in Section 6.1, and may also have direct access to the trust view, this is not designed to be used during normal operation. Initialization of trust assessments, trust validation and the trust view update is performed autonomously in the background, using passively collected local information (from past behavior and interactions) and the input of validation services and the reputation system for CA trust management (cf. Section 5).

The only user interaction during normal operation is that the relying entity provides the security level, he requires for the web site or web service he is about to open, as an input to trust validation. While an automatized decision on the security level by the determination of the class of application together with a predefined rule set would be desirable, this is out of scope of this work. Possible solutions can, for example, be based on content filtering (as also used to detect phishing sites [62]) or based on analyzing the type of entered data (cf. [44]). For now, the required security level can be given by the entity, using radio buttons that provide the different options for security levels from which an entity may choose as part of the browser's user interface.

Additional user interaction is only required as a fallback mechanism in case the validation services finally cannot provide a decision on the acceptance of an end entity certificate and trust validation outputs '$unknown$'. Then, the relying entity must decide upon acceptance. However, in this case, no experiences are collected for the involved CAs. Note that experience collection can be caught up later by putting the certificates on a watch list and adding the experiences after a later reconfirmation. As the lack of expertise makes user involvement problematic, the '$unknown$' case needs to be avoided whenever possible by the use of an adequate set of validation services. Further research on additional information sources and the optimization of the use of validation services is due to future work. Yet, given sufficient support, involving the user for decision making might be a viable approach, for example, when a bank provides his customers with further information about their certificates.

Apart from these cases no user interaction is required. In particular, relying entities do not actively provide their personal assumptions about the trustworthiness of obtained certificates, involved CAs or even the content of a web page and feed them back into CA-TMS. This is also one of the main differences to other user-centric approaches, like the Web of Trust [60], where users vote for the trustworthiness of provided content, or PGP [63], where users state their opinion about the trustworthiness of other users and the legitimacy of their public keys.

4.5 Trust view update

New information needs to be incorporated into the trust view to be available during future trust validations. Based on the output of the trust validation, either positive or negative experiences are collected for the involved trust assessments. Herein, it is important that only strictly new information is collected. Therefore, it is checked if a certificate, which is contained in the considered certification path, is evaluated for the first time based on the state of the trust view. We present the detailed algorithm in the following:

Input:

- A certification path $p = (C_1, ..., C_n)$ without intermediary self signed certificates
- A trust view View
- An output of the trust validation R
- A list of new trust assessments TL
- A Boolean value $v \in \{true, false\}$ that indicates whether C_n was validated by validation services or not.
- A list of validation services $VS = (vs_1, ..., vs_j)$ with possible outputs $R_i = vs_i(C) \in \{trusted, untrusted, unknown\}, 1 \leq i \leq j$ on input of a certificate C.
- *(optional)* A reputation system RS (as described in Section 5) which outputs recommended issuer trusts $rs(k, ca) = (\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$ on input of a pair (k, ca).

Output: The updated trust view.

The algorithm proceeds as follows:

1. If $R = unknown$ then return View
2. For $1 \leq i \leq n-1$ set k_i to the public key in C_i, set ca_i to the subject in C_i and set $TA_i = (k_i, ca_i, S_i, o_{kl,i}, o^{ca}_{it,i}, o^{ee}_{it,i})$ to the corresponding trust assessments.
3. *(optional)* If $(R = trusted \land v = true)$ then $\forall TA_i \in TL$ do:
   (a) Request $rs(k_i, ca_i) = (\tilde{o}^{ca}_{it,i}, \tilde{o}^{ee}_{it,i})$ from RS
   (b) If RS did not return *unknown* do:
   If $(i < n-1)$ set $o^{ca}_{it,i} = (0.5, 0, E(\tilde{o}^{ca}_{it,i}))$, else set $o^{ee}_{it,i} = (0.5, 0, E(\tilde{o}^{ee}_{it,i}))$.
4. If $R = trusted$ then
   (a) For $1 \leq i \leq n-1$ do
      i. If $C_i \notin S_i$ then add C_i to S_i
      ii. If $(i = n-1)$ then update $o^{ee}_{it,i}$ with a positive experience, else if $(TA_{i+1} \in TL)$ or $(C_{i+1} \notin S_{i+1})$ then update $o^{ca}_{it,i}$ with a positive experience.
      iii. If $TA_i \in TL$ then add TA_i to View.
   (b) Add C_n to View as trusted certificate.
5. If $R = untrusted$ then
   (a) Set $h = \{max(i) : TA_i \notin TL \text{ or the consensus of } (vs_1(C_i), ..., vs_j(C_i)) = trusted\}$.
   (b) For $1 \leq i \leq h-1$ do
      i. If $C_i \notin S_i$ add C_i to S_i
      ii. If $(TA_{i+1} \in TL)$ or $(C_{i+1} \notin S_{i+1})$ then update $o^{ca}_{it,i}$ with a positive experience.
      iii. If $TA_i \in TL$ then add TA_i to View
   (c) If $C_h \notin S_h$ add C_h to S_h
   (d) If $TA_h \in TL$ then add TA_h to View
   (e) If C_{h+1} is not an untrusted certificate in View then: if $(h < n-1)$ update $o^{ca}_{it,h}$ else update $o^{ee}_{it,h}$ with a negative experience.
   (f) Add C_{h+1} to View as untrusted certificate.
6. Return View

Given a new certificate that was evaluated as trusted, a positive experience is collected for the issuing CA. In case the certificate was evaluated as untrusted, a negative experience is collected. For the calculation of the trust value alone, every negative experience cancels out a positive one and vice versa. In many situations, negative experiences should have a bigger influence on trust than positive ones: a certificate that failed the evaluation should not be trusted less but not at all. To meet these concerns, the affected untrusted certificate is immediately added to the list of untrusted certificates, in addition to the collection of a negative experience for the CA that issued the certificate. Thus, the certificate will never be accepted in the future. Moreover, the actual impact of negative experiences on the final outcome of the trust evaluation can be controlled by using adequate security levels (cf. Section 4.4 and 6.1).

Example A simplified example of the evolution of the trust view is shown in Figure 1. It visualizes the experience collection process. Root CAs are denoted with R-CA, Sub CAs with S-CA. The arrows represent observed certificates.

(a) The system obtains the chain R-CA$_1$ $\rightarrow$ S-CA$_1$ $\rightarrow$ EE$_1$. The certificate of EE$_1$ is accepted. A positive experience is added to each involved CA.
(b) The chain R-CA$_1$ $\rightarrow$ S-CA$_2$ $\rightarrow$ EE$_2$ is obtained. The certificate of EE$_2$ is accepted. A positive experiences is added to each involved CA.

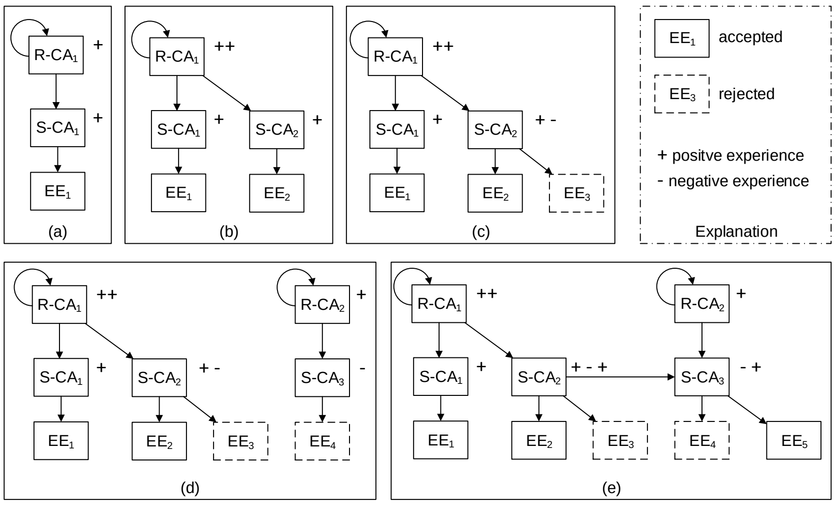

Fig. 1. Evolution of the trust view

(c) The chain R-CA$_1$ $\rightarrow$ S-CA$_2$ $\rightarrow$ EE$_3$ is obtained. The certificate of EE$_3$ is rejected. A negative experience is added to S-CA$_2$. However, the certification R-CA$_1$ $\rightarrow$ S-CA$_2$ was approved during prior observations, thus no negative experience is added to R-CA$_1$.
(d) The chain R-CA$_2$ $\rightarrow$ S-CA$_3$ $\rightarrow$ EE$_4$ is obtained. The certificate of EE$_4$ is rejected. Thus, the certificate R-CA$_2$ $\rightarrow$ S-CA$_3$ must be checked. Assuming its reconfirmation, a negative experience is added to S-CA$_3$, while a positive experience is added to R-CA$_2$.
(e) The chain R-CA$_1$ $\rightarrow$ S-CA$_2$ $\rightarrow$ S-CA$_3$ $\rightarrow$ EE$_5$ is obtained. The certificate of EE$_5$ is accepted. A positive experience is added to S-CA$_2$ and S-CA$_3$. R-CA$_1$ $\rightarrow$ S-CA$_2$ was evaluated during prior observations, no new experience is added.

Fixing the key legitimacy Different from the issuer trust, which might change over time, key legitimacy theoretically is constant once it is approved. From that point on, the issuer trust in superordinate CAs is of no further relevance. To consider this fact in the trust validation, key legitimacy is set $o_{kl} = (1, 1, 1)$ as soon as enough evidence for the key legitimacy of a trust assessment is available. We fix the key legitimacy after a CA's key was involved into several certification paths served by different web servers. This strategy is similar to multi path probing applied by certificate notaries. To realize the strategy, we introduce the parameter fix_{kl} and set $fix_{kl} = 3$, meaning that we fix the key legitimacy after collecting three positive experiences for the respective trust assessment (cf. Section 6.1 for an evaluation of the effects on the trust view for different parameter choices). Other approaches could be to fix the key legitimacy based on the number of certificates contained in the trust assessment or given multiple independent certification paths have been encountered.

Cleaning the trust view To prevent a continuous growth of the trust view and to allow the adaptation to current requirements (e.g., changing browsing behavior), a removal mechanism is integrated. A trust assessment TA is removed from the local trust view after a fixed time period has been passed since TA was last used within trust validation. The length of this time period can be implemented as a system parameter, e.g., one year. The determination of the optimal parameter setting is due to future work.

5 A reputation system for CA trust management

In the above sections, we have described how the trust view is established incrementally by querying notaries for any certificates as long as not enough locally collected information is available. We propose to integrate an additional reputation system to increase the amount of data that decisions are based on from local experiences to aggregated opinions from people with similar browsing behavior. While doing this, the trust views are kept minimal in order to adhere to the principle of least privileges, which is currently not followed in the Web PKI [13]. Therefore, an entity's requirements concerning CAs have to be considered during the aggregation of the proposed issuer trust. Otherwise, experiences collected for application uses completely irrelevant to the requesting entity might lead to unnecessarily high trust values in this entity's trust view. Furthermore, the reputation system is only queried after the relevance of a CA for an entity was approved due to the CA being part of a reconfirmed certification path (cf. Section 4.5). The purpose of the reputation system is to improve the bootstrapping of the trust view and to evolve an entity's trust view towards a stable state. When reaching a stable state, CA-TMS may work mainly autonomously, even for entities that only collect only few own experiences.

The remainder of this section is organized as follows. First, we discuss the architecture of the reputation system in Section 5.1 and its basic functionality in Section 5.2. Afterward, in Section 5.3, we discuss a strategy to select trust views for the computation of the recommended issuer trust. In the case of insufficient information on the reputation system's side, a service provider handover may be performed as described in Section 5.4. In Section 5.5, we discuss additional services that a reputation service provider may offer to its users. While being nonfunctional for the reputation system itself, these services provide an additional benefit that may attract entities to use the reputation system. Finally, we discuss privacy aspects that arise from uploading personal trust views to the reputation system.

5.1 Architecture

There are two main principles for building reputation systems [38], namely the centralized and the distributed architecture. But only the centralized one is suitable in the case of recommending issuer trust of CAs. In a distributed approach the participants are responsible to collect ratings directly from other participants. But in our setting, this requires another way of authenticating arbitrary Internet users in order to build so called identity trust, which is essential to distributed reputation systems [38]. However, requiring a pre-established authentication system in order to build a reputation system that shall enhance another authentication system would reduce the whole approach to absurdity.

Thus, we realize the reputation system using a centralized service provider (or, for scalability reasons, a network of service providers). The users register at the service provider and upload their trust views. Note that the authentication of a single service provider is not considered a problem. E.g., its certificate may be hard coded into the client software. Due to the registration, users can be re-identified when requesting a recommendation for the issuer trust of a CA with key k. The service provider aggregates the opinions from many trust views in his database to a recommendation and provides it to a requesting user (or the public).

Model To describe the reputation system we extend our model from Section 2.2. Recall that there exists an entity e_1 with trust view View and another entity e_2. e_1 establishes a TLS connection to e_2 and needs to decide if the connection is trustworthy. Additionally, there are other entities $u_1, ..., u_n$ (other Internet users) with trust views $\mathsf{View}_1, ..., \mathsf{View}_n$ and a network of service providers $sp_1, ..., sp_m$. The service providers are assumed to have pre-established trust relationships and are able to communicate securely, i.e. their keys are exchanged using an out of band channel. The network of service providers does not need to be complete, i.e., a service provider is not required to trust any other service provider. We assume that e_1 and $u_1, ..., u_n$ have registered at sp_1 and uploaded their trust views to the database of sp_1. In general, an entity can choose which service provider to use. Thus, each service provider has its own customer base and set of trust views. The clients' local trust views are regularly synchronized with the ones in sp_i's database. Figure 2 depicts the architecture.

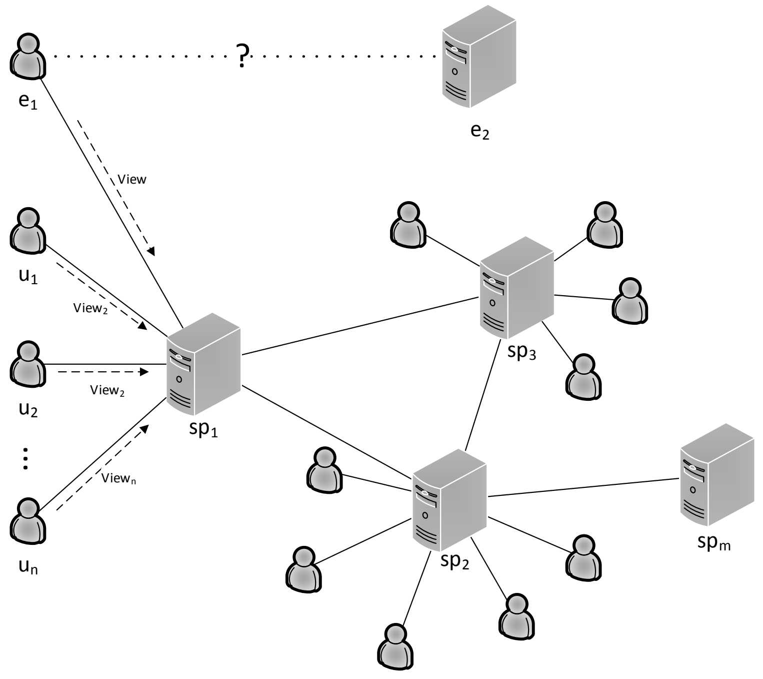

Fig. 2. Architecture of the reputation system

5.2 Functionality

The reputation system provides recommendations for the issuer trust assigned to a CA with key k. Whenever e_1 updates his trust view with a new trust assessment TA for CA ca and public key k, e_1 requests a recommendation for the associated issuer trust from sp_1. sp_1 aggregates the correspondent opinions for ca from j different trust views by applying the conflict aware fusion operator *cFUSION* (cf. Definition 2). In case sp_1 does not have information about ca, sp_1 forwards the request to other service providers he trusts.
The process is the following:

1. e_1 establishes a TLS connection to sp_1 and authenticates itself (e.g., by using a user name and password).
2. e_1 sends the pair (k, ca) to sp_1 using the secure connection.

3. Depending on e_1's trust view, sp_1 selects $j \geq 0$ trust views $\mathsf{View}_1, ..., \mathsf{View}_j$ from its database according to selection strategy S (see Section 5.3 for trust view selection strategies).
4. If $j > 0$ do
   (a) For $1 \leq i \leq j$ sp_1 extracts $o^{ca}_{it,i}$ and $o^{ee}_{it,i}$ for (k, ca) from View_i.
   (b) sp_1 aggregates the opinions on the issuer trust with the cFUSION operator $\hat{o}^{ca}_{it} = \hat{\oplus}_c(o^{ca}_{it,1}, ..., o^{ca}_{it,j})$ and $\hat{o}^{ee}_{it} = \hat{\oplus}_c(o^{ee}_{it,1}, ..., o^{ee}_{it,j})$.
5. If $j = 0$ (i.e., sp_1 has no information for (k, ca)) sp_1 forwards the request to another service provider he trusts. The other service provider responds with a recommendation $(\hat{o}^{ca}_{it}, \hat{o}^{ee}_{it})$ or with $unknown$. This step may be repeated or run in parallel for several service providers.
6. sp_1 responds to e_1 with either the aggregated issuer trust scores $(\hat{o}^{ca}_{it}, \hat{o}^{ee}_{it})$ or, if no recommendation is available, with *unknown*.

In the following we describe in detail how to choose trust views for the aggregation step and how to aggregate opinions to a single recommendation.

5.3 Trust view selection and trust aggregation strategy

We first present the selection and aggregation strategy. Afterward we show, how to apply clustering for pre-computation and efficiency improvements.

Trust view similarity weighting and cut off The aggregation of the recommended issuer trust should consider an entity's individual requirements. This cannot be achieved by simply averaging the respective opinions over all trust views in the service provider's database. The recommendation should be based on the trust views of entities that have comparable requirements as the requesting entity, namely entities with similar browsing behavior and similar security requirements. From the fact that CAs mostly work on certain domains [39] and the dependence of the trust views on the subjective browsing behavior as shown in [3], we follow that trust views reflect an entity's requirements in respect to the relevance of CAs.

Thus, we propose to weight the trust views to be aggregated based on their similarity and to cut off all opinions with a similarity below a certain lower bound b. Similarity of trust views can be measured with the Jaccard similarity index $J(A, B)$ defined in [11].

Similarity of trust views The Jaccard similarity index is a measure for the similarity of sets. Given two sets A, B, the Jaccard similarity index is defined as

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

where $|A|$ is the cardinality of A. Informally speaking, $J(A, B)$ is the number of common elements divided by the total number of elements contained in the two sets. Only considering the sets of TAs contained within trust views, the Jaccard similarity index can be applied to trust views as follows. Herein, we consider the TAs from different trust views as equal, if the contained CA name and key are identical. Then, $J(\mathsf{View}_1, \mathsf{View}_2) = \frac{n}{n_1+n_2-n}$, where n_i is the total number of TAs contained in View_i, $i \in \{1, 2\}$ and n is the number of TAs shared by the trust views. Note that the inclusion of certificates into the computation is omitted due to privacy issues (cf. Section 5.6).

Trust view selection process Let $TV_{(k,ca)}$ be the set of trust views in the service provider's database that contain a trust assessment for (k, ca) and let e_1 be the requesting entity with trust view View. Then the service provider chooses all trust views $\mathsf{View_i}$, $i = 1, .., j$ from $TV_{(k,ca)}$ for which the weight $w_i > b$ with $w_i = J(\mathsf{View}, \mathsf{View}_i)$. From these views, the service provider extracts the opinions on issuer trust $o^{ca}_{it,i}$ and $o^{ee}_{it,i}$ for (k, ca) and aggregates them using the cFUSION-operator (cf. Definition 2) with the corresponding weights w_i. Thereby, trust views that are more similar to the requesting entity's trust view influence the aggregated recommendation more than less similar trust views.

Cutting off trust views with weights below the bound b prevents trust views that are too different from View from being taken into account. The rationale behind this is that the cFUSION-operator only considers weights relatively. Thus, in case that solely trust views with a low Jaccard similarity are found, this would result in relatively high weights. Cutting off those trust views may come at the cost of not finding adequate trust views but prevents the recommendation of high issuer trusts, if the importance of a CA to an entity is not plausible. The definition of an optimal bound b requires a larger data set than currently available to us and is due to future work. A legitimate choice could be $b = 0.8$ (cf. Section 6.1 for exemplary comparisons of trust views). Besides that, the similarity weighting and cut of strategy provides protection against Sybil attacks which we discuss in Section 6.4.

One drawback of this strategy is the computational cost and scalability. To identify the j trust views for aggregation, the Jaccard similarity index needs to be computed for all trust views in the service provider's database that contain a trust assessment for (k, ca). This can be done more efficiently by clustering the trust views and then performing a local search within the cluster that is most similar to the one of the requesting entity.

Trust view similarity clustering Trust views can be clustered according to their similarity. The resulting clusters can be used to find similar trust views by only measuring a trust view's similarity to a few cluster centers. Even though the result will in general be less precise compared to computing the full set of similarities, it still contains "nearby" trust views. Clusters can be used to realize a pre-selection of trust views and realize the cut off of distant trust views more efficiently when the service provider's database contains many trust views. Now we explain how clustering is realized. Note that clustering can be done as a pre-computation within regular time intervals.

K-means clustering With K-means clustering and the Jaccard similarity index for trust views, κ clusters of similar trust views can be built. According to [25], the main steps of K-means clustering are:

- initially select κ random trust views as cluster centers
- repeat the following steps until cluster center convergence:
  - **assignment:** assign each trust view to cluster center for which the Jaccard similarity is maximal
  - **update:** reselect the cluster center within each cluster such that the arithmetic mean of Jaccard similarities of the new center with all other trust views in the cluster is maximized

K-means tends to make cluster sizes equal and each trust view is assigned to only one cluster. Equal sized clusters have the advantage of always providing a certain number of candidate trust views for the aggregation step, and the computational effort is bounded during request.

On the other hand, K-means clustering often fails finding the "natural" partitioning [33], where entities may belong to multiple groups and groups may have very different sizes and shapes. Furthermore, K-means clustering only finds local optima.

To escape local optima, K-means clustering is normally run several times with different initializations and for different sizes of κ. The outcomes are then compared using the arithmetic mean of similarities to the cluster centers. For the selection of the most suitable outcome, the criterion of a maximizing the arithmetic mean of similarities (which solely considered would lead to clusters of size one) and the criterion of adequate cluster sizes for aggregation need to be balanced.

Therefore, the parameter κ, which steers the number of clusters and thus their sizes, is chosen depending on the lower bound b to which trust views are accepted for the trust aggregation. Given b, we set $\kappa = 1/(1-b)$ as a first approximation. Given trust views were equally distributed within the set of possible trust views, this choice would lead to clusters where the minimum similarities to cluster centers were around b. Since trust views are not distributed equally, outliers with lower similarities will occur. If the number of outliers is above a certain limit, for example 10%, κ is increased in order to rerun K-means.

5.4 Service provider handover

In case the service provider sp_1 has no trust views to compute a recommendation for (k, ca) – either there is no trust view with a trust assessment for (k, ca) at all or none that meets the minimal similarity constraint b – he can request a recommendation from other service providers.

sp_1 queries other service providers $sp_2, ..., sp_m$ he trusts for their recommendation. If more than one trusted service provider answers with a recommendation, i.e. aggregated opinions on issuer trust for (k, ca), the responses are aggregated using the cFUSION operator with equal weights. Querying other service providers is transparent to the requesting entity. If all service providers $sp_2, ..., sp_m$ respond with $unknown$, sp_1 also answers with $unknown$ to the requesting entity.

In order to enable $sp_2, ..., sp_m$ to locally perform the aggregation of opinions to a recommendation as described in Section 5.3, sp_1 hands over the requesting entity's trust view. To protect the entity's privacy, the trust view can be shortened by all end entity certificates.

Note that if sp_1 utilizes clustering, sp_1 could hand over the center of the cluster to which the entity's trust view is assigned. While this provides a stronger privacy protection, it comes at the expense of precision during the computation of the weights during the aggregation step and it cannot be guaranteed that the minimal similarity constraint b is met.

5.5 Additional services

A centralized service provider can provide additional complementary services to its users. On the one hand, the services can incentivize more users to submit their trust views and thereby indirectly enhance the accuracy of the reputation system due to the availability of a larger set of data for calculation of recommendations. On the other hand, the services provide additional benefits in itself. We describe potential services that provide a surplus for users.

Backup Uploading and regularly updating a trust view results in a copy of this trust view on the service provider's side. This–in fact–is a backup as trust views are linked to user accounts. Providing the option to download one's own trust view is a simple backup solution for all users.

Migration to other devices Trust views develop incrementally and are highly user specific. Thus, switching to new devices (or a new web browser) should not lead to a loss of the trust view. Instead of having to start with a new trust view, the backup service provides all functions necessary to migrate trust views from one device to another as long as both are under control of the same user.

Synchronization between devices Whenever a user concurrently uses several devices (or several web browsers), there is an interest to access the same set of trust information, i.e., the same trust view, on all devices. With the possibilities already provided by the backup feature, synchronization of a user's trust view between all possessed devices is possible. Furthermore, by implementing incremental synchronization and filtering by use cases, a more sophisticated approach can be taken.

Revocation monitoring A service provider who knows the trust view of a user can monitor the revocation status of the contained certificates and inform the user about revocations. While revocation is considered problematic [32] in many ways, this can resolve availability issues of revocation information, as these do not have to be available during connection establishment, but are obtained on a regular basis from the service provider.

5.6 Privacy issues

Trust views contain the certificates of CAs and end entities that a user has had contact with. The data from trust views can be used to profile users and their web browsing habits at least concerning services that use secure connections. Thus, trust views are sensitive data in terms of privacy which needs to be considered when this data is not exclusively maintained locally but uploaded to a service provider.

Despite the fact, that a reputation system requires the data in plain and reveals trust information to other users, privacy protection is possible. Recommendations are in general aggregated knowledge of several of its users. The recommendations are furthermore not linked to the source of information and trust views are never revealed as a whole to the public. Thus, a recommendation at most reveals the trust in a single CA and therefore is not considered privacy critical.

Besides that, not the complete trust views are required for the reputation system, but only the trust assessments. As CAs in general sign certificates for arbitrary web services, the privacy criticality of this part of the trust views is limited. As mentioned above, this can be utilized during service provider handovers. To even protect this information from being known to the service provider of a user, the parts of the trust view that reveal which services a user consumes (namely the set of trusted and untrusted end entity certificates) can be stored encrypted with a user-specific key.

6 Evaluation

In this section we evaluate the system. We first summarize the different system parameters and show their influence on the trust views by simulating the system behavior based on the data sets collected in a previous study [3]. Afterward, we present the security gains and discuss attacks and defense mechanisms regarding trust management systems.

In the study, data about browsing histories from different entities where collected. From the histories, the web services (hosts) accessed via TLS connections were extracted along with the respective certification paths. These paths are available sequentially ordered based on the date when they where first accessed. This sequentially ordered paths allow us to simulate the evolution of the respective trust views. Note that in the simulation, the collection of experiences is limited to positive experiences. Furthermore, querying the reputation system cannot be simulated. Due to only collecting positive experiences, the resulting opinions on the issuer trusts form an upper bound for the actually derived trustworthiness in real life applications. For the simulations we used the twenty histories that contain data for at least one year.

6.1 Parameters and system behavior

The system behavior is controlled through different system parameters. We distinguish between general parameters and parameters that are only relevant for the optional reputation system.

General parameters

Parameter n The parameter n of CertainTrust opinions is the number of the expected average number of experiences for a statement. It is a system-wide parameter. We propose $n = 10$, which means that after collecting ten experiences for one of the opinions, its certainty becomes 1.

Due to its impact on the certainty of an opinion, the parameter n of CertainTrust influences the development of the expectation of opinions during the course of collecting experiences. This is shown in Figure 3 for different values of n. While n has only little influence on the expectation during the collection of the first three to four experiences, n significantly influences the number of required positive experiences to reach expectation values approaching the upper bound of 1. That means, n, in conjunction with the security levels, can be used to adjust how fast CAs are considered fully trustworthy, while n concurrently has only minimal influence on the CAs that are approaching minimal security levels. Thus, increasing n mainly means shifting CAs from the group of fully trustworthy CAs to medium trustworthy CAs while the group of CAs that reach the minimal security level remains unaffected.

Security levels The expectation value of CertainTrust opinions is continuous. Thus, the required security level l for each application could also be chosen as any number between 0.5 and 1. However, we propose to assign applications to three distinct classes as already discussed in Section 4.4. Doing so makes the system clearer and easier to use for potential users, as it reduces the complexity to decide which security level to require for an application.

We choose three security levels to which a user may assign applications as $l_{min} = 0.6$, $l_{med} = 0.8$, and $l_{max} = 0.95$. A CA for which the expectation of the derived key legitimacy for issued host certificates exceeds l_{max} is considered to be fully trustworthy (medium or minimally trustworthy respectively).

Given a complete key legitimacy of the CA's key and $n = 10$, at least one positive experience has to be collected to reach the minimal security level for certificates issued by the CA. The medium security level requires three positive experiences while the maximum security level requires seven. For this choice, it can be seen that increasing n, e.g., up to $n = 30$, would have no influence on when a CA is considered minimally or medium trustworthy, but twelve instead of seven positive experiences would be required for the CA to be considered fully trustworthy.

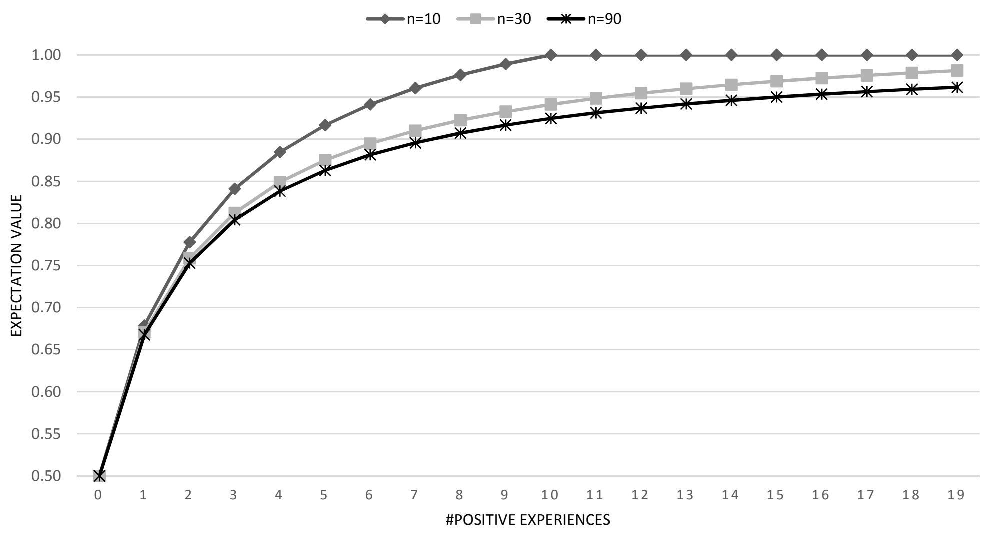

Fig. 3. The influence of n on the development of the expectation of a CertainTrust opinion.

Parameter fix_{kl} In Section 4.5, we proposed to fix the key legitimacy of a CA's key after observing a CA's key within fix_{kl} certification paths. After the fixation of the key legitimacy, the certification path has no further influence on the evaluation of certificates issued by the CA. Then we rely solely on the experiences made with the CA directly. We propose to fix the key legitimacy after three positively evaluated encounters, i.e., set $fix_{kl} = 3$.

Fixing the key legitimacy effects the development of the trust view. Similar to n, increasing fix_{kl} mainly reduces the number of fully trustworthy CAs, while the number of medium and minimal trustworthy CAs grows. This is shown in Figure 4. It depicts exemplary for one trust view how the distribution of hosts associated with CAs reaching the different security levels changes for values from $fix_{kl} = 1$ up to $fix_{kl} = 80$. The distribution is measured after observing 604 different TLS-enabled hosts.

The explanation for the effects yielded by fixing the key legitimacy is that the expectation for issued certificates is not lowered by the trustworthiness of superordinate CAs (which is due to transitively lowering the key legitimacies along the certification path). Thereby, this mainly affects the distribution of CAs among the reached security levels, while the effect on the number of hosts that are associated to CAs not reaching any security level is rather low.

The value of fix_{kl} also has influence on how fast a CA achieves its previous security level after its key was changed. Directly after the key renewal, the derived key legitimacy of signed end entity keys will be low, as the key legitimacy of the CA's key is computed based on the path from the root to the CA's (new) certificate. But after fix_{kl} encounters of the CA's key, the previous security level is reestablished, i.e. the system solely relies on the experiences collected for the CA in question.

Stereotyping and parameter $maxF$ Stereotyping is a means to derive trust for newly observed CAs based on experiences collected for other CAs. This, on the one hand, allows to trust in CAs never observed before. Thus the number of required reconfirmations is reduced. On the other hand, the anticipated trustworthiness of the CA is not based on the CA's behavior anymore and information gained from stereotyping should not be overestimated to prevent threats. Threats may evolve when a less trustworthy CA profits from the reputation

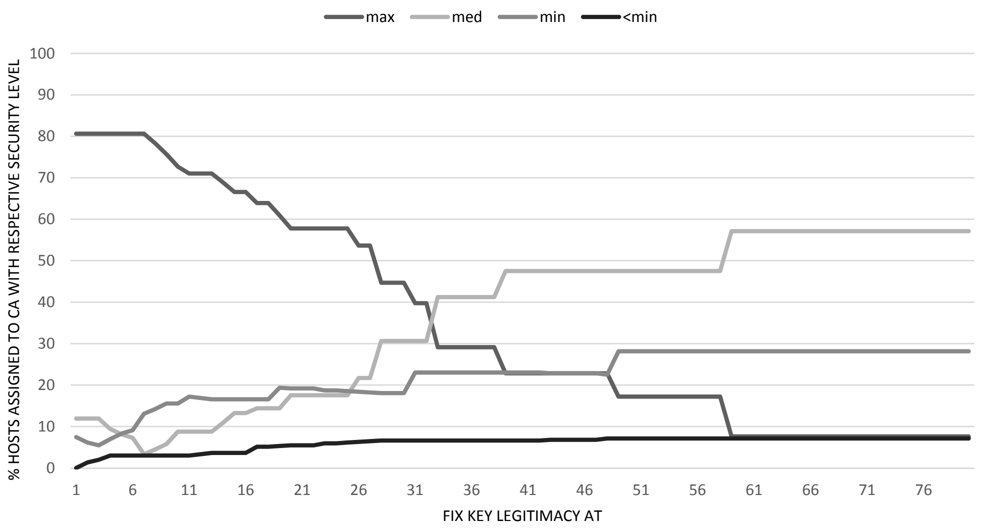

Fig. 4. Percentage of total hosts that are assigned to a CA with the security level max: $l \geq 0.95$, med: $0.95 > l \geq 0.8$, min: $0.8 > l \geq 0.6$ and <min: $l < 0.6$ for the derived key legitimacy of certificates issued to end entities for different values of fix_{kl}. For one exemplary trust view.

of its siblings. Therefore, we limit the influence of stereotyping by selecting the parameter $maxF = 0.8$. This limits the certificates that are directly trusted due to stereotyping to low security applications.

The effects of stereotyping and fixing of the key legitimacy The effects of stereotyping and fixing of the key legitimacy is shown in Figures 5-7. The figures show how the percentage of hosts, for which a reconfirmation is required, develops over the course of the evolution of an exemplary trust view. This is shown for different required security levels.

Using only stereotyping has practically no effect if the required security level is l_{max} (see Figure 5: the rates for the strategies *basic* and *stereotyping* are nearly identical), while fixing of the key legitimacy lowers the rate of reconfirmations by 50%. In the combined strategy, both mechanisms strengthen each other's effects. For a required security level of l_{med} (cf. Figure 6), stereotyping performs better as the basic strategy. However, the strengthening effect in the combined strategy is not observable. Figure 7 shows the strongest effect of stereotyping, being equally effective as fixing the key legitimacy. In this case, the effects accumulate. The effects of stereotyping are weaker than the effects of key legitimacy fixing at first, but after the encounter of several hosts, both curves approach each other. For stereotyping to strike, experiences for siblings must be available first, which is not the case in the early states of a trust view.

The difference in the functioning of the two mechanisms is that fixing the key legitimacy mainly affects known CAs and therewith new host certificates issued by the known CAs, while stereotyping affects newly observed CAs and the certificates issued by those. To sum up, using both strategies concurrently, significantly reduces the number of hosts for which a reconfirmation is required.

Figure 8 shows the evolution of an exemplary trust view, concerning the distribution of hosts to minimal, medium, and maximal trustworthy CAs. The parameter setting thereby is as follows: $l_{max} = 0.95$, $l_{med} = 0.8$, $l_{min} = 0.6$, $maxF = 0.8$, $fix_{kl} = 3$, and $n = 10$.

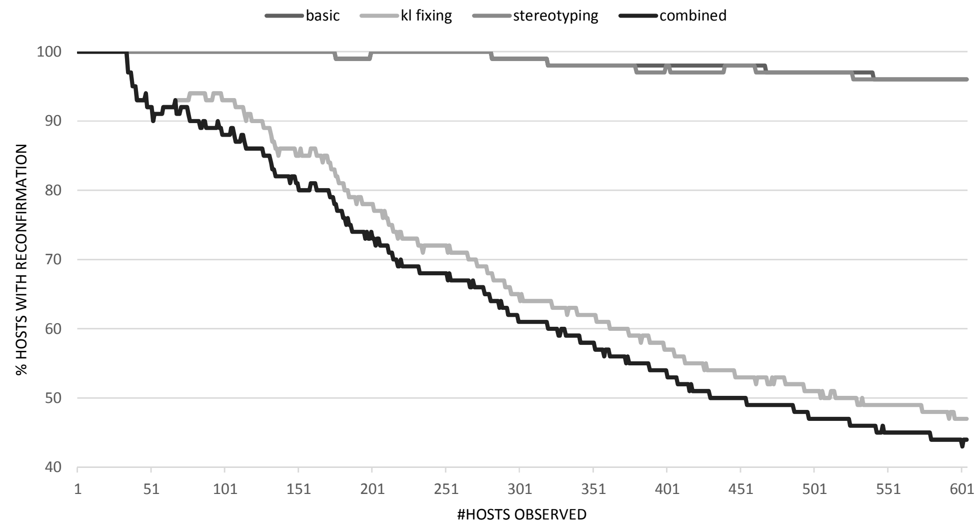

Fig. 5. Percentage of observed hosts for which a reconfirmation was required given a required security level $l = 0.95$ considering the different strategies: *basic*: no fixing of key legitimacy or stereotyping; *kl fixing*: fixing of key legitimacy with $fix_{kl} = 3$ and no stereotyping; *stereotyping*: stereotyping used but no fixing of key legitimacy; *combined*: both fixing of key legitimacy and stereotyping used.

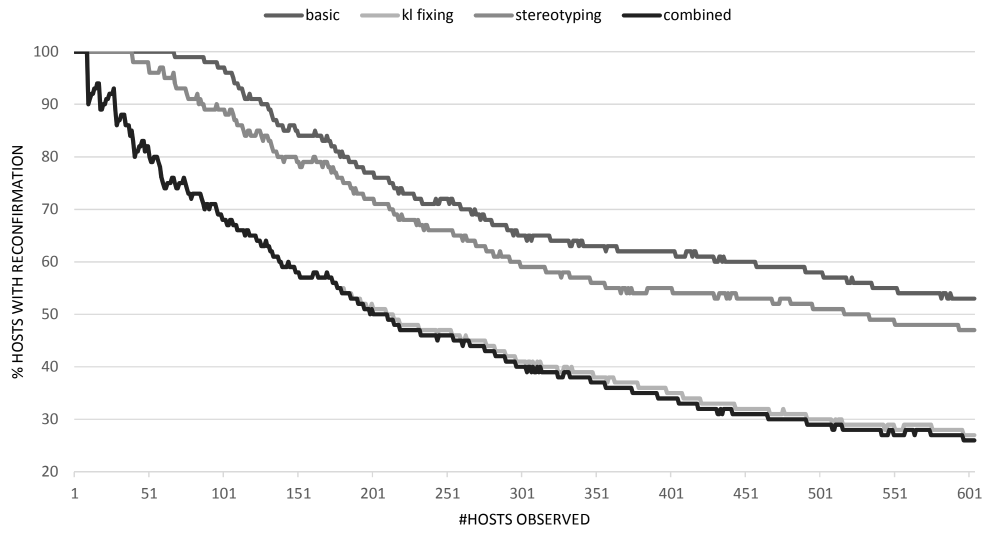

Fig. 6. Percentage of observed hosts for which a reconfirmation was required given a required security level $l = 0.8$ considering the different strategies: *basic*: no fixing of key legitimacy or stereotyping; *kl fixing*: fixing of key legitimacy with $fix_{kl} = 3$ and no stereotyping; *stereotyping*: stereotyping used but no fixing of key legitimacy; *combined*: both fixing of key legitimacy and stereotyping used.

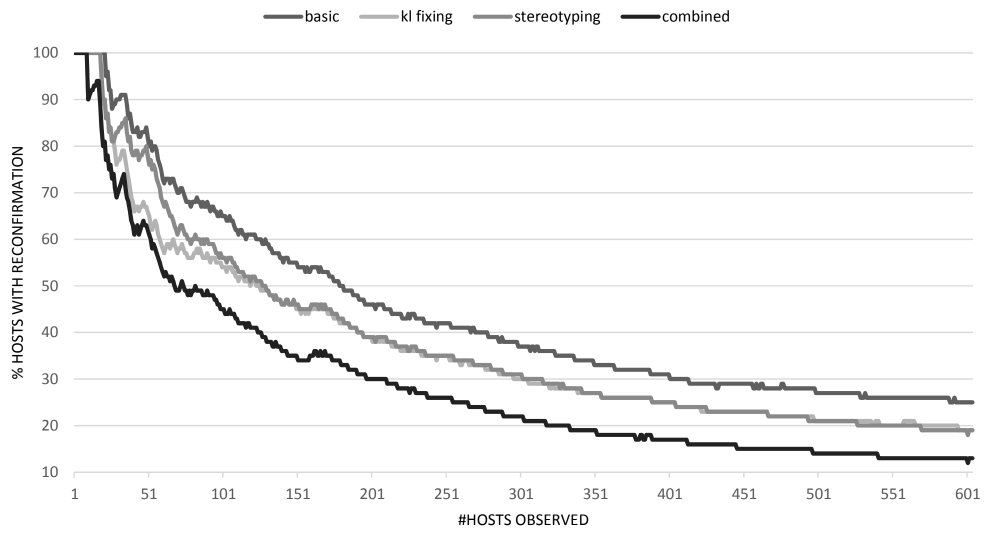

Fig. 7. Percentage of observed hosts for which a reconfirmation was required given a required security level $l = 0.6$ considering the different strategies: *basic*: no fixing of key legitimacy or stereotyping; *kl fixing*: fixing of key legitimacy with $fix_{kl} = 3$ and no stereotyping; *stereotyping*: stereotyping used but no fixing of key legitimacy; *combined*: both fixing of key legitimacy and stereotyping used.

The curve denoted $<$*min* depicts the percentage of hosts assigned to CAs that did not reach any security level. The curve drops below 10% after the observation of around 120 different hosts with the corresponding certification paths. It can be interpreted as an approximation for the marginal rate of reconfirmations, assuming that the minimum trustworthiness of a CA is adequate in most cases. The approximated marginal rate of reconfirmations for applications requiring the maximal security level can be computed as one minus the percentage of hosts assigned to fully trusted CAs. For medium security applications analogously as one minus the percentage of fully and medium trusted CAs.

System behavior for different trust views Now, we show the system behavior for the different trust views for the following parameter values: $l_{max} = 0.95$, $l_{med} = 0.8$, $l_{min} = 0.6$, $maxF = 0.8$, $fix_{kl} = 3$, and $n = 10$.

Figure 9 shows the distribution of hosts to the CAs, for which the derived key legitimacy for issued end entity certificates reach the security levels l_{max}, l_{med} or l_{min}. The plotted data shows the distribution for the twenty analyzed trust views after completely simulating the history information. The trust views are ordered by the amount of different hosts found in the history. On the X-axis, the number of hosts for the respective trust views is depicted. A growing number of hosts leads to a growing percentage of CAs being trusted with a maximum of 85% of hosts assigned to fully trusted CAs. The percentage of hosts assigned to untrusted CAs continuously drops. As Figure 10 shows, this does not come from a dramatic growth in the number of fully trusted CAs. The percentage of CAs that are able to issue fully trusted certificates reaches a maximum of 28% of the total number of CAs contained in the corresponding trust view. In summary, only a rather small number of fully trusted CAs (we found a maximum of 29 CAs) issues the majority of the certificates an entity observes.

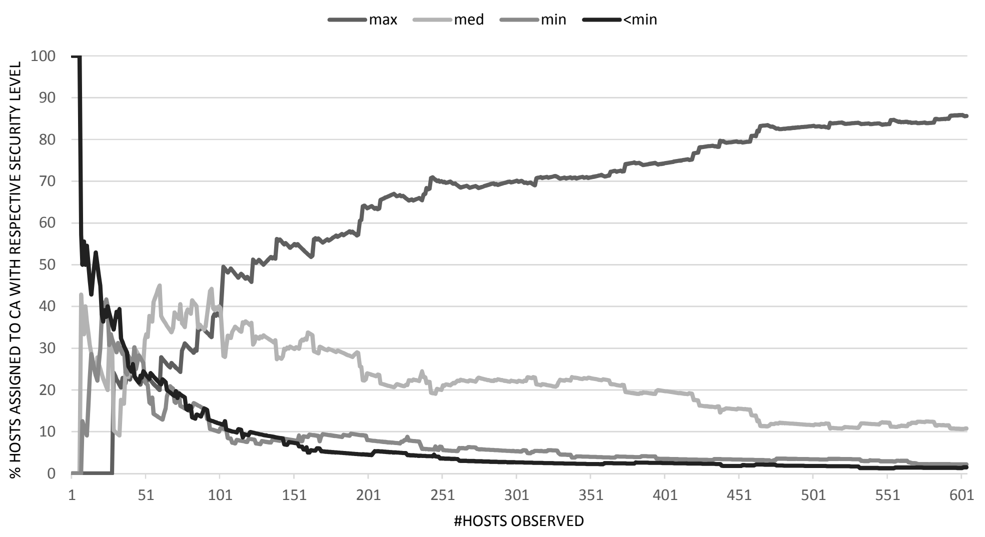

Fig. 8. Changes of the percentages of total hosts that are assigned to CAs with the security level max: $l \geq 0.95$, med: $0.95 > l \geq 0.8$, min: $0.8 > l \geq 0.6$, and <min: $l < 0.6$ for the derived key legitimacy of certificates issued to hosts, during the course of the evolution of a trust view.

While for trust views based on histories that contain only few hosts, the distribution of hosts to trusted CAs is rather chaotic, it seems to stabilize around 100 observed hosts. This also holds for the percentage of CAs that reach the respective security levels. There is a clear rationale: Firstly, for trust views with only few hosts only few experiences can be collected. Secondly, the number of CAs grows strictly under-proportional to the number of CAs. The ratio $\frac{\#CAs}{\#hosts}$ hereby drops from 1.6 to 0.17 in our data sets, meaning that for each CA more experiences are collected on average. This means that trust views perform good for entities that access many different hosts (good in the sense that entities mainly rely on their local data and do not require to reconfirm most of the certificates). For other types of entities, most certificates need to be reconfirmed (while the absolute number is rather low). In such cases, a reputation system as described in Section 5 can help in categorizing the CAs such entities observe. We note that each host certificate only needs to be reconfirmed once due to certificate pinning.

The complete numbers for the twenty analyzed trust views can be found in Tables 1 and 2. The number of CAs that do not reach any security level ($l < 0.6$) in Table 2 only contains CAs that actually do sign end entity certificates. CAs only issuing Sub CA certificates are not included.

Reputation system parameters

Trust view similarity bound b We evaluated the Jaccard similarity measure on the twenty trust views derived from the twenty analyzed histories. The average similarity of one view to all others thereby lies between 0.25 and 0.52, while the maximum similarity that we found between two trust views was 0.69. Our data set is not large enough to determine the best value for the similarity bound b, from which on trust views are considered for the computation of recommendations. However, we believe that selecting $b = 0.8$ is suitable—given a large enough database of trust views.

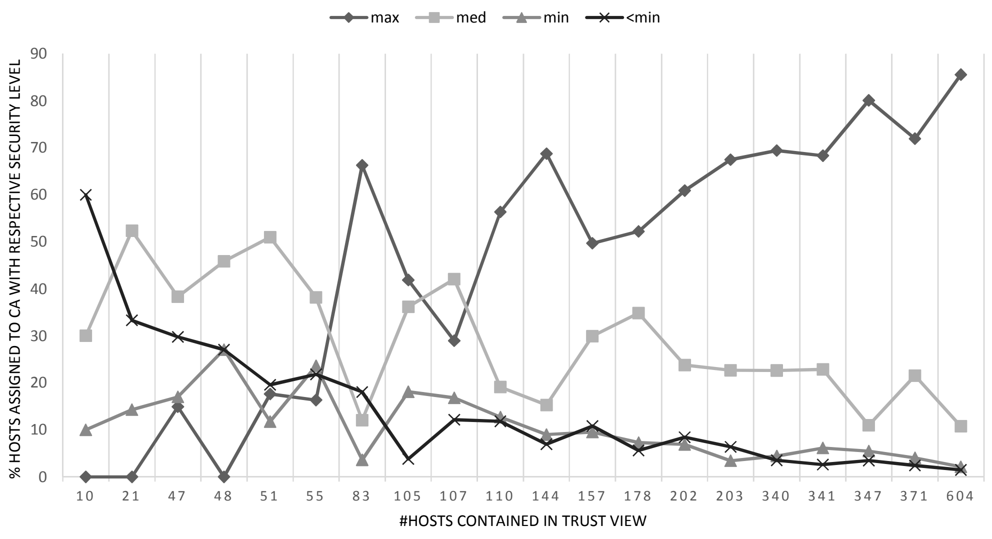

Fig. 9. Percentage of total hosts that are assigned to CAs with the security level max: $l \geq 0.95$, med: $0.95 > l \geq 0.8$, min: $0.8 > l \geq 0.6$, and <min: $l < 0.6$ for the derived key legitimacy of certificates issued to end entities.

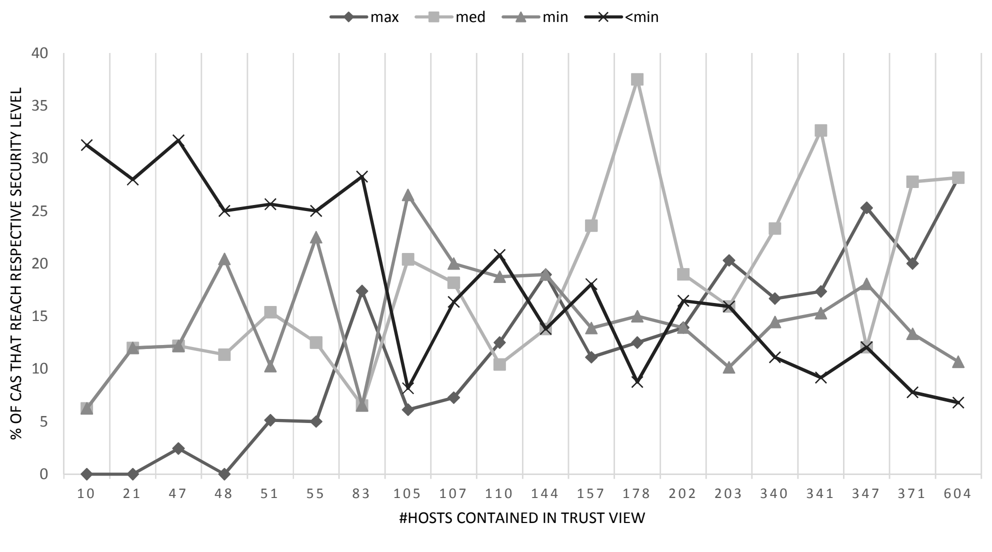

Fig. 10. Percentage of total CAs that reach the security level max: $l \geq 0.95$, med: $0.95 > l \geq 0.8$, min: $0.8 > l \geq 0.6$, and <min: $l < 0.6$ for the derived key legitimacy of certificates issued to end entities.

Trust View	#hosts	#hosts assigned to CAs with security level			
		$l \geq 0.95$	$0.95 > l \geq 0.8$	$0.8 > l \geq 0.6$	$l < 0.6$
1	10	0	3	1	6
2	21	0	11	3	7
3	47	7	18	8	14
4	48	0	22	13	13
5	51	9	26	6	10
6	55	9	21	13	12
7	83	55	10	3	15
8	105	44	38	19	4
9	107	31	45	18	13
10	110	62	21	14	13
11	144	99	22	13	10
12	157	78	47	15	17
13	178	93	62	13	10
14	202	123	48	14	17
15	203	137	46	7	13
16	340	236	77	15	12
17	341	233	78	21	9
18	347	278	38	19	12
19	371	267	80	15	9
20	604	517	65	13	9

Table 1. Numbers of hosts and their distribution to CAs with different security levels for different trust views.

Trust View	#CAs	#host signing CAs	#CAs trusted for security level			
			$l \geq 0.95$	$0.95 > l \geq 0.8$	$0.8 > l \geq 0.6$	$l < 0.6$
1	16	7	0	1	1	5
2	25	13	0	3	3	7
3	41	24	1	5	5	13
4	44	25	0	5	9	11
5	39	22	2	6	4	10
6	40	26	2	5	9	10
7	46	27	8	3	3	13
8	49	30	3	10	13	4
9	55	34	4	10	11	9
10	48	30	6	5	9	10
11	58	38	11	8	11	8
12	72	48	8	17	10	13
13	80	59	10	30	12	7
14	79	50	11	15	11	13
15	69	43	14	11	7	11
16	90	59	15	21	13	10
17	98	73	17	32	15	9
18	83	56	21	10	15	10
19	90	62	18	25	12	7
20	103	76	29	29	11	7

Table 2. Numbers of CAs and host signing CAs and their distribution to security levels for different trust views.

As explained in Section 5.3 the initial choice for κ directly depends on b. With $b = 0.8$, κ is initially set to $\kappa = 5$.

6.2 Attacker model

We summarize the attacker's goal, capabilities, and limitations.

Attacker goal The attacker targets at TLS connections established between entities and web servers. The goal of the attacker is to inject a malicious certificate for which he controls the private key during key agreement in order to be finally able to eavesdrop, monitor or manipulate the secured communication. The goal is achieved if the maliciously generated certificate is not detected and thus accepted by the target entity.

Attacker capabilities The attacker is active. Thus, he is able to intercept and alter the communication between two entities, e.g. he acts as a man-in-the-middle. In the evaluation, a distinction is made between permanent attackers that can intercept all communication of an entity and time-limited attackers that can only intercept communication channels during a fixed time frame.

Additionally, the attacker can obtain malicious certificates issued on behalf of at least one CA of the Web PKI. The malicious certificates are obtained either by compromising a CA's private key and using it to issue certificates, or by a CA failure. Note that self-signed certificates are unsuitable for attacks as they will be detected during standard path validation.

Besides the attacker specific capabilities, the attacker has all capabilities of regular entities. For example, the attacker can register at a reputation service and upload trust views.

Attacker limitations The relying entity's system under consideration is assumed not to be compromised (which does not mean that there do not exist compromised or malicious user systems at all). Thus, the attacker is unable to access or manipulate the locally stored trust view. We do not consider relying entities that change their own trust view maliciously, which indeed would only be to their own disadvantage. Such malicious behavior (of other entities) becomes only relevant when considering the upload of trust views to the reputation system and is discussed under Sybil attacks in Section 6.4.

The reputation system itself is assumed not to be compromised, i.e., the attacker is unable to arbitrarily manipulate the database of the reputation system or its computation processes. Even further, it is assumed that the communication between relying entity and the corresponding servers of the reputation system is secure. Intrusion detection and attacks on user systems are separate fields of security research and independent from this paper. Furthermore, we assume that the attacker does not manipulate the validation services to provide false reconfirmations.

The attacker is not capable of breaking the cryptographic algorithms. In particular this means, once a secure channel has been established the attacker cannot eavesdrop on the content of the communication or manipulate the sent data undetectably. Also, standard security mechanisms of the Web PKI are assumed to be intact. For example, the detection of a maliciously issued certificate results in countermeasures like revocation and blacklisting. The time period between compromise and detection is called gray period. The actions of an attacker are limited to the gray period, as a malicious certificate would be detected afterward by standard means.

6.3 Protection against PKI attacks

In this section, we evaluate how CA-TMS reduces the attack surface of the Web PKI given CA-TMS is not manipulated by the attacker. Attacks against CA-TMS are discussed separately in Section 6.4.

When the relying entity uses CA-TMS, trust validation must succeed for a presented certificate chain. Otherwise, validation services are queried and a potential attack is detected. Trust validation can only succeed if the certificate was either used before or if the issuing CAs are contained in the entity's trust view. Furthermore, those CAs must be considered sufficiently trustworthy for the required security level of the application. Thus, the attacker must compromise a CA with sufficiently high issuer trust in order to be successful. Attacking a specific group of entities requires the compromise of a CA with a sufficiently high issuer trust in each of the group member's trust views. The same holds when attacking a specific service. In this case, the relevant set of trust views are those of the group of service users.

Therefore, the attacker needs to identify a CA that is sufficiently trusted by all targeted entities. Otherwise, the attacker risks an immediate detection of the compromise when the validation services are triggered. However, as trust views are specific to individual entities and not publicly visible, it is hard to identify such a sufficiently trusted CA. Even if this is possible, it is questionable if the attacker can purposefully compromise this CA.

Generally speaking, by the use of CA-TMS, an attacker can hardly exploit accidental CA failures. The possible damage is reduced due to the limitation of the number of attackable entities accompanied by the increased compromise detection probability. Furthermore, with CA-TMS, the damage a compromised CA may cause highly depends on the CAs visibility in the certification business. The result of using CA-TMS is a much more natural setting than each existing CA being equally critical.

Reduction of the attack surface The overall attack surface is reduced by CA-TMS by limiting the number of trusted CAs. We measure the effectiveness of the system by adapting the metric of Kasten et al. [39], which measures the attack surface as $AS = \sum_{ca \in CAs} dom\,[ca]$. Therein, $dom\,[ca]$ is the number of domains that a given CA is allowed to sign. CAs describes the set of all CAs which are part of the Web PKI. We extend the metric to:

$$AS(\mathsf{View}) = \sum_{ca \in CAs} (b_1^{ca} \cdot dom_{max} + b_2^{ca} \cdot dom_{med} + b_3^{ca} \cdot dom_{min})$$

with

$$b_1^{ca} = \begin{cases} 1 & \text{if for } ca: \ E(o_{it}^{ee}) \geq l_{max} \\ 0 & \text{else} \end{cases},$$

$$b_2^{ca} = \begin{cases} 1 & \text{if for } ca: \ E(o_{it}^{ee}) \geq l_{med} \\ 0 & \text{else} \end{cases},$$

$$b_3^{ca} = \begin{cases} 1 & \text{if for } ca: \ E(o_{it}^{ee}) \geq l_{min} \\ 0 & \text{else} \end{cases},$$

where dom_{max}, dom_{med}, and dom_{min} are the respective numbers of domains for which the relying entity requires a maximal, medium or minimal security level. The input View represents the trust view of the entity.

Herein, $dom_{max} + dom_{med} + dom_{min} = dom$, which describes the total number of validly signed domains, i.e., for which a valid TLS certificate exists. Note that for simplicity, we do not parametrize dom by the respective CA as we do not consider the restriction of domains for which a CA is allowed to issue certificates.

We base our calculations on the number of 1590 CAs found by Durumeric et al. [13]. Currently, all CAs are trusted for signing certificates for any domain, thus $AS = 1590 \cdot dom$, which equals the attack surface computed with the original metric.

The metric enables us to relatively quantify the reduction of the attack surface resulting from the use of CA-TMS with the above specified parameters. As the distribution of the domains to dom_{max}, dom_{med}, and dom_{min} depends on the relying entity's preferences, the exact attack surface is also specific to the entity and furthermore depends on its specific trust view. We do not have data about the distribution for the data sets analyzed in Section 6. Thus, we do the comparison for the three extreme cases where either l_{max}, l_{med} or l_{min} is assigned to all domains. If we apply the metric to our data sets, the relative attack surface is simply the quotient of CAs in the trust view that can issue certificates for a given security level (cf. Table 2) divided by the total number of CAs. For l_{max}, the relative attack surface lies between 0 and 0.018, for l_{med} between 0 and 0.036 and for l_{min} between 0.001 and 0.043. In any case, the surface is reduced by more than 95%. The minimum numbers clearly come from the least evolved trust views where nearly no CAs are trusted. This comes at the cost of querying validation services whenever new hosts are observed. However, even in the most evolved trust view, where 604 different TLS-enabled hosts were accessed and the approximate marginal rates of reconfirmation (cf. Figure 8, Section 6.1) are below 15%, 5%, and 2% for the different security levels l_{max}, l_{med} or l_{min} shows the enormous security increase by using CA-TMS. Nevertheless, the overhead by querying validation services is kept in an acceptable range.

Limitations While trust views can significantly reduce the attack surface, past experiences are no guarantee for correctness. Trust views do not protect CAs from being compromised. If a CA, for which many positive experiences were collected, suddenly fails, the relying entity may still falsely rely on a maliciously issued certificate. On the other hand, it is also possible that a connection is falsely evaluated not to be trustworthy. The behavior lies in the nature of basing decisions on incomplete information.

Furthermore, trust in the key legitimacy of a service provider's key is different from the trustworthiness of the service provider itself. The trustworthiness of a service provider comprises, for example, the quality of the web page and its content. The latter is not addressed by CA-TMS and requires additional mechanisms like the Web of Trust [60] or commercial web page ratings, e.g., Norton SafeWeb [49] or McAfee SiteAdvisor [46]. However, such mechanisms require authentication, which is again achieved via authentic public keys.

6.4 Protection against computational trust attacks

This section is focused on attacks against specific components of CA-TMS. Attacks are categorized into two groups: either attacks on manipulating a target entity's local trust view or attacks aiming towards manipulating the external reputation system. The latter one ultimately leads to a manipulation of local trust information on the target entity's side, too. Following from the above attacker model, the intention behind these attacks is to finally increase the attacker's success probability when he attacks the entity's communication with a forged certificate. Note that the reputation system is only queried when a new trust assessment is initialized *after* the legitimate use of the CA's key has at least been reconfirmed once. Thus, the reputation system cannot be employed by an attacker to introduce additional trust assessments into the target entity's trust view. Following the analysis framework from [27], our analysis identifies the attack vectors based on the separation into formulation, calculation, and dissemination components of CA-TMS.

Formulation resembles the reputation metric and sources of input. A CA's reputation is stored as opinion, which is updated by positive or negative experiences. The information source is either direct, namely an observed certification path, or indirect via the reputation system. Thereby, the information source for the reputation system are trust views uploaded by its users. Thus, an attacker can positively or negatively influence the trust in a CA, either by directly serving manipulated certification paths or by influencing the recommendations served by the reputation system. The reputation system itself may be influenced via uploading manipulated trust views.

Calculation concerns the algorithms that derive trust from the input information. In CA-TMS, these algorithms are deterministic and executed either locally or, concerning a recommendation, centrally by the reputation system. Thus, as the system of the relying entity and the reputation system are assumed not to be compromised, an attacker cannot influence the calculation other than by injecting manipulated input data.

Finally, *dissemination* concerns all transfer of data between system components. As communication between user systems and the reputation system as well as between different service providers is secure, the attacker cannot manipulate the communication. He may indeed block or disturb the communication and thus, perform a denial-of-service (DoS) attack on the reputation system. In this case, CA-TMS falls back on local information. As a validated certification path is always checked by validation services before the reputation system is queried, a DoS does not introduce a direct threat. As DoS attacks can be detected, the blocked information can be resent later. DoS attacks are therefore of limited relevance for security and thus not considered further.

Attack vectors Following from the analysis, the possibilities to manipulate and influence the trust evaluation of CA-TMS under the given assumptions is limited to injecting false input data. We now explain the attack vectors an attacker might use to inject false data either directly into the entity's trust view or indirectly via the reputation system, and explain the respective protection mechanisms. We also discuss social engineering which may serve as a auxiliary attack vector to support other attacks.

Certification path manipulation To directly inject information into the trust view, the attacker needs to inject manipulated certification paths into the target entity's communication during the connection establishment. In order to inject positive experiences (for a CA of which the attacker controls the key), he must inject the CA's certificate into certification paths such that trust validation succeeds. Note that validation services could also provide false information. However, this was excluded for the analysis as the security of validation services is outside the scope of this paper.

To inject the CA's certificate, the attacker can either replace the whole path, by generating a new certificate for the web service with the CA's key, or by issuing a certificate for one of the intermediary CAs contained in the original path sent by the web server. Therefore, this certificate replaces only the part from the Root CA to the newly issued Sub CA's certificate. In both cases, standard path validation succeeds.

The first option is exactly from what CA-TMS shall protect and was discussed in Section 6.3. The attacker then may only succeed if the compromised CA is already trusted by the relying entity. This means that this attack vector may only be used to increase the trust in CAs that are already trusted. The second path manipulation option will not be detected as the end entity certificate remains unchanged. Yet, this way the attacker can only increase the corresponding issuer trust for issuing CA certificates.

Injecting forged certificates in order to generate negative experiences is impossible for CAs the attacker does not control, as he cannot generate certificates in the name of CAs of which he cannot access the keys.

Sybil attacks When targeting the reputation system, the attacker needs to inject manipulated trust views into the database of the reputation system to finally manipulate the recommendations. This is done using the scheme of a Sybil attack. Performing a Sybil attack means the attacker forges or controls a large amount of entities and acts on behalf of them. Also, a large amount of entities that act in a coordinated manner when uploading manipulated trust views resembles a Sybil attack. Whereas a single entity acting maliciously only has limited influence, as its opinions would be overruled by a majority of honest entities.

Whenever an attacker is able to register itself with several forged identities at the reputation system (or the attacker can control the systems of already registered entities), the attacker can upload manipulated trust views in their name to the reputation system. As the reputation system generates its opinions based on the trust views of its users, the attacker can influence the recommendations either negatively or positively by adding a suitable trust assessment for the targeted CA to the trust views he controls. If the attacker controls a large enough fraction of a reputation system's user base, the attacker effectively controls the content of the recommendations generated by the reputation system.

There are several defense mechanisms to prevent Sybil attacks. Countering Sybil attacks usually includes *making the registration of new users costly* to prevent attackers from generating and registering fake identities. Popular mechanisms include user authentication upon registration, a registration fee or computationally complex registration processes, e.g., CAPTCHAs. As our reputation system requires a registration of its users anyway, these are adequate measures.

Furthermore, the selection strategy for the computation of recommendations using similarity weighting (cf. Section 5.3) provides protection against Sybil attacks (cf. [56] for a similar approach). When the reputation system is requested to calculate a recommendation on the issuer trust for a CA, only those trust views are selected that are similar to the one of the requesting entity. However, the requesting entity's trust view in general is unknown to the attacker. A trust view uploaded by an attacker is only by chance considered during the calculation. This makes it difficult for an attacker to generate and submit trust views to manipulate the recommendation without knowing the target entities trust view. The trust view is only communicated over secured connections. Thus, gathering information about trust views requires observing the TLS traffic of the target entity or sophisticated social engineering attacks. Even if the attacker manages to tailor the manipulated trust views for one target entity, the overall success is limited.

Moreover, proactive techniques can be implemented to protect reputation systems against Sybil attacks. The upload of a large amount of trust views within a certain time interval or sudden changes of a CA's issuer trust within trust views can be statistically detected as shown in [61]. Besides that, Sybil attacks can be lessened when the reputation system considers only trust views or trust assessments of a certain minimal age. The presented techniques significantly increase the costs of an attack, as they increase the time span during which the attack is to be executed. This helps to bridge the gray period and after the associated certificate has been revoked or blacklisted, the attack is anyway without effect.

Social engineering In general, social engineering means an attacker tries to manipulate other entities such that they behave differently than they would in the absence of a social engineering attacker.

The direct influence of social engineering attacks on the reputation of a CA is limited, as relying entities do not manually set the key legitimacy or issuer trust assigned to a CA. In CA-TMS, user involvement is minimal. Relying entities only have to make their own decision on the acceptance of a certificate if their trust view does not contain sufficient information and if all queried validation services respond with $unknown$. However, if a relying entity manually accepts a certificate, no experiences are collected for the CAs. The only possibility to directly influence the trust view of a relying entity is to lead the relying entity to web pages whose certificates were legitimately signed by the CA controlled by the attacker. This can either introduce additional CAs into the trust view or result in the (legitimate) collection of additional experiences for a CA.

Another possibility for an attacker is to influence a relying entity to setting the required security level l to a low value, which would increase the probability for the attacker's certificate to be accepted. Therefore, CA-TMS should recommend reasonable values and warn the user when changing the security level. Furthermore, automatic detection of the required security level according to a specific rule set in conjunction with the content of a web page is an interesting future research direction.

Additionally, an attacker could try to gain information about the user's browsing behavior, for example by interviewing the user to find out some of his interests. This in fact would help the attacker during a Sybil attack to generate trust views similar to the one of the relying entity. Anyway, these sorts of social engineering attacks are limited to single relying entities and require a lot of preparation.

Attacker goals and defenses The previous subsection discussed the attack vectors that an attacker can use to manipulate trust views. In this section, specific attacker goals, their possible realization, and how they apply to CA-TMS are discussed.

Self-promoting Self-promoting describes actions of the attacker towards making him or a CA under his control appear more trustworthy. This in fact is the attacker goal with highest relevance as it increases the success probability of the attacker when finally issuing malicious certificates to attack the secured communication.

A self-promoting attacker can approach his goal either by injecting manipulated certification paths or by a Sybil attack on the reputation system. As discussed in Section 6.4, certification path manipulation only works for the issuer trust concerning the issuance of CA certificates. This might subsequently allow an attacker to issue CA certificates for which the key legitimacy is high. However, he cannot influence the issuer trust concerning the issuance of end entity certificates, which in fact is required to benefit from the attack. This is only possible for CAs that are already trusted making the attack needless.

To use a Sybil attack, the attacker must overcome the above described defense mechanisms. Additionally, as the reputation system is only queried when a trust assessment is newly initialized, the CA controlled by the attacker must be new to the targeted entity. On the other hand, due to the use of validation services, the CA must be legitimately observed first, which in fact can hardly be steered by the attacker for multiple entities. Even for a single entity, this requires social engineering coordinated with the Sybil attack and perfect timing.

Slandering In opposition to a self-promoting attack, slandering aims at lowering the reputation of a specific CA. As the attacker does not directly benefit from decreased trust in CAs, he might only aim at disturbing the proper functioning of CA-TMS. As described in Section 6.4, an attacker cannot inject malicious certificates on behalf of CAs he does not control, thus he cannot utilize the attack vector of manipulated certification paths to inject negative experiences.

The only possibility is to use a Sybil attack on the reputation system. However, he must overcome the defense mechanisms and may only influence newly initialized trust assessments on the relying entity's side. During the bootstrapping of a trust view, this has a certain impact, but afterward, the attack is of limited relevance. Furthermore, the attack in the worst case increases the number of required reconfirmations. In this case, local experiences are collected, which leads to a fade out of the influence of the manipulated recommendation.

Whitewashing Whitewashing describes the approach of an entity with negative reputation to re-appear under a new, clean identity.

An attacker that controls a CA with negative reputation might want to give this CA a new identity to issue certificates that will appear trustworthy. However, whitewashing does not apply to CA-TMS. Newly observed certificates are not automatically trusted and thus, an attacker gains no advantage from whitewashing. Moreover, whitewashing is prevented by standard PKI mechanisms as for a CA to re-appear under a new identity, its new CA certificate either needs to be added to the root stores or needs to be certified by some CA which is already part of the Web PKI.

7 Conclusion & future work

We have presented CA-TMS, a user-centric CA trust management system for the Web PKI based on trust views. The trust view maintains a minimal set of trusted CAs and furthermore assigns different ratings to each CA, such that trust decisions can be made depending on the context. Thus, the risk of relying on a malicious certificate can be governed by the assignment of adequate security levels to the applications. This enables more restrictive trust decisions for critical applications like e-banking, where security is more important and less restrictive rules for less security critical applications. These rules can be adapted to the relying entity's risk profile.

By the local and user-centric management of trusted CAs instead of relying on global trust settings, the attack surface of the Web PKI is reduced. We have discussed the different parameters of CA-TMS and presented a suitable parameter setting. Our evaluation based on authentic browsing histories shows that CA-TMS allows a reduction of the attack surface to less than 5% of its original size. This prevents attacks induced by CA failures and compromises. Attacks employing CA keys not contained in the relying entity's trust view are detected. The security analysis shows, that the majority of attacks based on CA failures and compromises can be prevented. Attacks against the trust management itself have been shown to require sophisticated planning on the attackers side, while the attacker only has incomplete information to base his attacks on. Furthermore, attacks need to be mounted over a certain time period. This has two effects. The probability of detection is increased and second, the increased attack complexity helps to bridge the gray period until a compromise is detected.

Notably, the core functionality of CA-TMS can be implemented on top of the existing infrastructure without its alteration and can be implemented locally on the relying entity's side. Solely the optional reputation system requires additional service providers. For the reconfirmation of certificates, CA-TMS falls back on notarial solutions. Our performance evaluation shows, that the frequency of occurrence of this fall back mechanism fades out the more local experiences are collected. This continuously reduces the overhead and possible delays. For entities that only collect few own experiences, the reputation system is an important addition, while certificate pinning prevents recurring reconfirmations.

A specific strength of CA-TMS is its extensibility which allows the addition of more information sources and leaves room for further improvements of the reconfirmation and data collection processes. To conclude, we state that CA-TMS does not open additional vulnerabilities because it is designed as an addition and not as a replacement of the Web PKI. This renders the system ready to use. The experimental use of the system can then also help to collect data and further investigate on improvements and optimal or alternative parameter settings.

Ongoing and future research We are currently implementing CA-TMS. The implementation will allow us to quantify the overhead and delays introduced by the additional trust computation and certificate checks. The minimization of delays during page loading is long known to be an important success factor for acceptance [1]. Besides that we will carry out user studies to achieve a good usability. A detailed analysis using a larger set of trust views accompanied by long term field tests will most likely reveal options for fine-tuning the system parameters, especially regarding the reputation system.

Regarding the extensibility of our system, we will investigate in the addition of further information sources. A promising extension might be combining local information with expert recommendations based on different indicators of trustworthiness as, e.g., derived from the evaluation of CA policies [58]. Furthermore, we aim at the flexible integration of arbitrary validations services, requiring methodologies to evaluate and combine different response types. Also, the incorporation of complementary approaches like CAge [39] is promising to further reduce the attack surface. On the other hand, the development of feedback and reporting mechanisms when anomalies are detected yields potential for future research.

Besides careful design in terms of security and robustness of our system, the prevention of false or maliciously inserted trust views into the reputation system is still an open field of research. Also, measures for privacy protection of the users need further research. Promising directions involve homomorphic cryptography and zero knowledge proofs.

Bibliography

[1] Akamai - Press Release. Akamai Reveals 2 Seconds as the New Threshold of Acceptability for eCommerce Web Page Response Times, 2009. `http://www.akamai.com/html/about/press/releases/2009/press_091409.html`, visited March 2014.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, 1996.

[3] 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*, pages 600–605, September 2013. Extended verison available at: `http://eprint.iacr.org/2013/275`.

[4] J. Braun, F. Volk, J. Buchmann, and M. Mühlhäuser. Trust Views for the Web PKI. In S. Katsikas and I. Agudo, editors, *Public Key Infrastructures, Services and Applications*, volume 8341 of *Lecture Notes in Computer Science*, pages 134–151. Springer Berlin Heidelberg, 2014.

[5] C. Burnett, T. J. Normal, and K. Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 241–248. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[6] Carnegie Mellon University. Perspectives Project. `http://perspectives-project.org//`, visited July 2012.

[7] D. W. Chadwick and A. Basden. Evaluating trust in a public key certification authority. *Computers & Security*, 20(7):592–611, 2001.

[8] W. T. H. Choice. SSL/TLS in a post-prism era. `https://wiki.thc.org/ssl#OtherIncidents`, visited December 2013.

[9] Comodo. The Recent RA Compromise. `http://blogs.comodo.com/it-security/data-security/the-recent-ra-compromise/`, visited November 2011.

[10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. RFC 5280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), 2008.

[11] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009.

[12] T. Dierks and E. Rescorla. RFC 5246 – The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), 2008.

[13] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. In *Proc. 13th Internet Measurement Conference (IMC '13)*, Barcelona, Spain, 2013. to appear.

[14] P. Eckersley and J. Burns. The (Decentralized) SSL Observatory. Invited talk at 20th USENIX Security Symposium, August 2011.

[15] The EFF SSL Observatory. `https://www.eff.org/observatory`.

[16] C. Ellison and B. Schneier. Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[17] C. Evans, C. Palmer, and R. Sleevi. Public Key Pinning Extension for HTTP. Internet-Draft, 2013.

[18] FOX IT. Black Tulip - Report of the investigation into the DigiNotar Certificate Authority breach, 2012. `http://www.rijksoverheid.nl/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update.html`.

[19] D. Gambetta. Can we trust trust? In *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, 1988.

[20] P. Gutmann. PKI: it's not dead, just resting. *Computer*, 35(8):41 – 49, August 2002.

[21] P. Gutmann. *Engineering Security*. Draft, 2013. Book draft available online at `http://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf`.

[22] h-online. Flame – oversights and expertise made for windows update worst case scenario. `http://h-online.com/-1614234`, visited July 2012.

[23] h-online. Attack on Israeli Certificate Authority. `http://h-online.com/-1264008`, visited November 2011.

[24] S. M. Habib, S. Ries, S. Hauke, and M. Mühlhäuser. Fusion of opinions under uncertainty and conflict – application to trust assessment for cloud marketplaces. In *TrustCom 2012*, pages 109–118, 2012.

[25] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[26] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 workshop on New security paradigms workshop*, NSPW '09, pages 133–144, New York, NY, USA, 2009. ACM.

[27] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1–31, Dec. 2009.

[28] P. Hoffman and J. Schlyter. RFC 6698 – The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), 2012.

[29] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. 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, pages 427–444, New York, NY, USA, 2011. ACM.

[30] J. Huang and D. Nicol. A calculus of trust and its application to PKI and identity management. In *IDTrust '09*, pages 23–37, New York, NY, USA, 2009. ACM.

[31] ICSI. The ICSI Certificate Notary, 2013. `http://notary.icsi.berkeley.edu/`.

[32] ImperialViolet. Revocation doesn't work. `https://www.imperialviolet.org/2011/03/18/revocation.html`, visited December 2013.

[33] A. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010.

[34] A. Jøsang. An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'99). The Internet Society*, 1999.

[35] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9:279–311, 2001.

[36] A. Jøsang. Fission of opinions in subjective logic. In *12th International Conference on Information Fusion (FUSION '09)*, pages 1911–1918. IEEE, 2009.

[37] A. Jøsang and R. Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[38] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43:618–644, 2007.

[39] J. Kasten, E. Wustrow, and J. Halderman. Cage: Taming certificate authorities by inferring restricted scopes. In A.-R. Sadeghi, editor, *Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 329–337. Springer Berlin Heidelberg, 2013.

[40] A. Langley. Further improving digital certificate security. `http://googleonlinesecurity.blogspot.de/2013/12/further-improving-digital-certificate.html`, visited December 2013.

[41] B. Laurie, A. Langley, and E. Kasper. RFC 6962 – Certificate Transparency. RFC 6962 (Experimental), 2013.

[42] S. Magin and S. Hauke. Towards engineering trust systems: Template-based, component-oriented assembly. In *Proceedings of the Eleventh Annual International Conference on Privacy, Security and Trust (PST2013)*, 2013.

[43] M. Marlinspike. Convergence. `http://convergence.io/`, visited July 2012.

[44] M.-E. Maurer, A. D. Luca, and S. Kempe. Using data type based security alert dialogs to raise online security awareness. In *SOUPS*, page 2, 2011.

[45] U. M. Maurer. Modelling a Public-Key Infrastructure. In *Proceedings of the 4th European Symposium on Research in Computer Security: Computer Security*, ESORICS'96, pages 325–350. Springer, 1996.

[46] McAfee. SiteAdvisor. `http://www.siteadvisor.com/`, visited March 2014.

[47] D. H. Mcknight and N. L. Chervany. The meanings of trust. Technical report, University of Minnesota, 1996.

[48] Microsoft Security Response Center. Security Advisory 2798897 , 2012. `http://technet.microsoft.com/en-us/security/advisory/2798897`.

[49] Norton. Safe Web. `https://safeweb.norton.com/`, visited March 2014.

[50] PSYC. Certificate Patrol. `http://patrol.psyced.org/`.

[51] S. Ries. Extending bayesian trust models regarding context-dependence and user friendly representation. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1294–1301, New York, NY, USA, 2009. ACM.

[52] S. Ries, S. M. Habib, M. Mühlhäuser, and V. Varadharajan. Certainlogic: A logic for modeling trust and uncertainty (short paper). In *TRUST 2011*, pages 254–261. Springer, 2011.

[53] S. Ruohomaa, L. Kutvonen, and E. Koutrouli. Reputation management survey. In *Seventh International Conference on Availability, Reliability and Security (ARES 2007)*, pages 103–111, 2007.

[54] C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against SSL. Technical report, Indiana University Bloomington - Center for Applied Cybersecurity Research, 2010.

[55] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *Proceedings of the 18th Conference on USENIX Security Symposium*, 2009. available online at `http://static.usenix.org/event/sec09/tech/full_papers/sunshine.pdf`.

[56] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 1–1, Berkeley, CA, USA, 2006. USENIX Association.

[57] A. S. Wazan, R. Laborde, F. Barrère, and A. Benzekri. A formal model of trust for calculating the quality of x.509 certificate. *Security and Communication Networks*, 4(6):651–665, 2011.

[58] A. S. Wazan, R. Laborde, F. Barrère, and A. Benzekri. The x.509 trust model needs a technical and legal expert. In *ICC*, pages 6895–6900, 2012.

[59] G. A. Weaver, S. Rea, and S. W. Smith. A computational framework for certificate policy operations. In F. Martinelli and B. Preneel, editors, *Public Key Infrastructures, Services and Applications*, volume 6391 of *Lecture Notes in Computer Science*, pages 17–33. Springer Berlin Heidelberg, 2010.

[60] WOT. Know which websites to trust. `https://www.mywot.com/`, visited March 2014.

[61] Y. Yang, Y. Sun, S. Kay, and Q. Yang. Securing rating aggregation systems using statistical detectors and trust. *Information Forensics and Security, IEEE Transactions on*, 4(4):883–898, 2009.

[62] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 639–648, New York, NY, USA, 2007. ACM.

[63] P. R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.