Trust Views for the Web PKI

Johannes Braun, Florian Volk, Johannes Buchmann and Max Mühlhäuser

Technische Universität Darmstadt/CASED
Hochschulstraße 10, 64283 Darmstadt, Germany
{jbraun|buchmann}@cdc.informatik.tu-darmstadt.de,
florian.volk@cased.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. In this paper, we present the concept of *trust views* to manage variable trust levels for exactly those CAs actually required by a relying entity. This reduces the set of trusted CAs and minimizes the risk to rely on malicious certificates issued due to CA failures or compromises.

1 Introduction

The Web PKI is one of the largest and most important cryptographic systems. The core of the Web PKI is the ecosystem of CAs that are responsible for the issuance and the maintenance of SSL certificates. These certificates are issued to web service providers and are used in the SSL/TLS protocols. 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, where privacy, confidentiality, and integrity are often indispensable.

However, the Web PKI fails in many points to provide the desired security [7, 9,10]. One serious problem is that the Web PKI does not scale with the enormous size of the Internet. For the sake of interoperability (i.e., as much legitimate web service certificates as possible should be verifiable) the number of CAs, which are fully trusted by default in current browsers and operating systems, has continuously been growing over the past. Currently, there are approximately 1.500 trusted CAs [6]. As each of these trusted CAs can sign certificates for any web service or domain, trusting a single malicious CA, i.e., one that is in fact not trustworthy, can break the whole Web PKI's security. An adversary, who is in possession of a fake certificate that was issued by 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 malicious or defective CA increases. Several security incidents in the last time clearly show that this is more than just a hypothetical threat [5, 9, 11, 12].

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 users at risk until the detection of the threat. As explained, the risk grows proportional to the number of CAs a user trusts.

In this paper, we propose a new approach to reduce this risk by reducing the number of trusted CAs to those that are really required by the users. Recent experiments with browser histories of different users have shown that on average, a user only depends on a small subset of CAs. The size of which lies in the range of 10% of the CAs available and trusted by default in the Web PKI [1]. 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 might be sufficient or not to consider a connection to a web service as secure.

To achieve this, we present the concept of *trust views* that serve as a local and user dependent knowledge base for trust decisions. We present the mechanisms for the establishment and the management of the trust view. We implement learning processes and define decision rules by employing computational trust models. The real trustworthiness of CAs is approximated by a subjective probabilistic trust value. Our approach allows an adaptation to the requirements of the user and automated trust decisions based on defined decision rules. Our system focuses on applicability, thus it only uses data which is already available or is collected over time. However, our system is open to be extended with additional information sources.

The paper is organized as follows. Section 2 describes the Web PKI and our security model. In Section 3 we introduce computational trust and present related work. Afterward, in Section 4 we present the trust view concept. We describe challenges and how a trust view is modeled. Then we describe the initialization mechanisms for trust views and give the relevant algorithms for trust validation and the update of the trust views. In Section 5 we evaluate our approach and discuss limitations. We end with a conclusion and future work in Section 6.

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 the knowledge of 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 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 [4]. Whenever a relying entity contacts a web server and successfully establishes an SSL/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 web servers. This way, a hierarchical structure is created. The chain of certificates starting with a Root CA's certificate 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* [4].

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 [20, 23, 33]. First, the relying entity must be convinced of the key legitimacy of the CA's key that was used to sign the certificate. Second, the relying entity must trust the CA to issue trustworthy certificates which 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. 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 an SSL/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 SSL/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.

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

1. C_n=C
2. 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 the concept of *trust views* and explain how this enables 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 Computational Trust

Computational trust is a means to support users in making decisions under uncertainty, e.g., under incomplete information. Jøsang defines *decision trust* in [19]:

> *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 [26] 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*.

There exists an ambilateral mapping between the experience space and the opinion space by parametrizing a Bayesian probability density function with the amount of positive and negative experiences. For details, see [25]. In this paper, trust information is collected in the experience space but the opinion space is used to combine trust statements about different CAs. Opinions can be updated with newly collected positive or negative experiences by mapping the opinion into the experience space, adding the new experience to either the number of positive or negative experiences and mapping those back to the opinion space.

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 [26]:

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:*

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

The CertainLogic AND-Operator is commutative.

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.

3.2 Related Work

The multitude of problems and disadvantages of the currently deployed Web PKI is described by well known researchers [7,9,10]. Monitoring of the Web PKI reveals its enormous size and shows that indeed malpractices are common [6,14].

Many attempts exist to circumvent the problems imposed by possible CA failures and thus to enhance Internet security. Certificate pinning (e.g., [8,24]) means that relying entities store certificates of formerly accessed websites. Based on the *trust on first use approach*, it implies that a possible adversary must be present during the first connection establishment to the website. 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). Notarial solutions [2,16,21] maintain databases containing formerly observed certificates and can be queried to reconfirm the authenticity of a specific certificate, sometimes also involving consensus decisions of several independent notary servers. In this approach, trust is deferred from the CAs to a majority of notaries. Central instances come with availability and scalability problems. Also, privacy protection issues and delays due to the communication overhead are clear disadvantages.

The enhancement of PKI with trust computation has been proposed by many researchers. The CertainTrust model and CertainLogic [26] used by us are equivalent to the Beta Reputation System and Subjective Logic, both by Jøsang et al. [17,18], 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. [19] and Ruohomaa et al. [27]. Jøsang proposes an algebra for trust assessment in certification chains in [20] but mainly addresses trust networks similar to PGP [33]. Huang and Nicol [15] 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 and we do not expect any entity to pay for a certificate that includes a low trust value. Different certificate classes like domain validated (DV) or extended validation (EV) can be indicators for such trust values, but in our opinion these are not sufficient for trust evaluation.

Other researchers base trust evaluation in CAs on their policies and the adherence to those [3,29]. This requires policy formalization [3,29,31] for automated processing. Such formalized policies are not provided by the 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 [30].

Our solution builds on the techniques of previous works, however there are several fundamental differences. We combine different mechanisms and use them as building blocks to solve separate subproblems. The novelty thereby is to locally and user-specific limit the number of trusted CAs to those the user actually requires. Different from pure pinning and notarial solutions the CAs in this user-specific set have different trust levels and might even be fully trusted depending on the context. Thus our solution does not require an additional check of each (new) certificate and provides a trade-off between overhead and solely relying on CAs. While we make use of established computation models for trust evaluation, we base it on local experiences of the user instead of using recommendations of trust values within certificates or the evaluation of certificate policies and expert opinions. Thus, our system can work autonomously and only requires notarial reconfirmation, when not enough local experience has been collected so far. Furthermore, the management of local experiences guarantees, that independent from the CA's global reputation it is not trusted without an additional check, if the CA has never been observed by the user before. This aims at also protecting the user from malfunctions of CAs that in general follow good security practices but are actually irrelevant for the user.

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-user 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 a user 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 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 [13, 28].
2. **Incomplete information on CA processes**: data on the quality of a CA's certification process might be incomprehensible or not available at all.
3. **Incomplete information on user requirements**: in general, the web services that a user will contact in the future are unknown and therefore also the required CAs 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 $(k, ca, S, o_{kl}, o_{it})$, 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 (key legitimacy of k).
- o_{it} is an opinion. It represents the trust of e_1 in ca to issue trustworthy certificates (issuer trust in ca, when using k).

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 $TA = (k, ca, S, o_{kl}, o_{it})$ 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:

- $k = \mathit{public\ key}$
- $ca = \mathit{CA\ name}$
- $S = \{C\}$
- $o_{kl} = (1, 1, 1)$ if the CA is a Root CA, else $o_{kl} = \mathit{unknown}$.
- $o_{it} = (0.5, 0, 0.5)$ if no prior information is available about the issuer trust of the CA. Else if for $1 \leq i \leq n$ there are trust assessments $TA_i \in \mathsf{View}$ with $C_i \in S_i$, where the issuing CA of C_i is equal to the issuing CA of C, then $f = (\sum_{i=1}^{n} E(o_{it,i}))/n$ and $o_{it} = (0.5, 0, f)$.

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 changed to complete as soon as enough evidence has been collected. We discuss this in Section 4.5.

The issuer trust $o_{it} = (0.5, 0, 0.5)$ reflects that no experiences have been collected and that the CA may either be trustworthy or not. If the new CA is certified by a CA that certified several other CAs, 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.

Optimally, further information is collected for initialization. Our system is open for such extensions. Further information can be gathered from policy evaluation as, e.g., proposed by Wazan et al. [29, 30]. 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 [1], it takes a certain time until enough experiences are collected such that the system may operate autonomously. 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 for details). A possibility for such a bootstrapping procedure is to scan the browsing history. From the history, the services that are accessed via *https* 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 a 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 $p = (C_1, ..., C_n)$
- 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.
- A required certainty $rc \in [0; 1]$. rc is selected by e_1 and represents on how much previous information the decision must be based to be accepted.
- 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 list $TL$.
   (c) For $1 \leq i \leq n-1$ set $o_{kl,i}$ to the key legitimacy of $k_i$ and $o_{it,i}$ to the issuer trust assigned to $k_i$ in View.
   (d) Set $h = \{max(i) : o_{kl,i} = (1,1,1)\}$
   (e) Compute $o_{kl,n} = (t,c,f) = o_{it,h} \wedge o_{it,h-1} \wedge \cdots \wedge o_{it,n-1}$
   (f) Compute the expectation $exp = E(o_{kl,n})$
   (g) If $exp \geq l$ then $R \leftarrow trusted$
   (h) If $exp < l$ and $c \geq rc$ then $R \leftarrow untrusted$
   (i) If $exp < l$ and $c < rc$ 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 $R_c = cons(R_1, ..., R_j)$ to the consensus on $(R_1, ..., R_j)$, then $R \leftarrow R_c$.
   (j) Update View. (See Section 4.5 for details.)
4. Return $R$
```

According to previous works [20,23,33], 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 : $o_{kl} = o_{kl,CA} \wedge o_{it,CA}$. 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, $o_{kl,n} = (t,c,f) = o_{it,h} \wedge o_{it,h-1} \wedge \cdots \wedge o_{it,n-1}$ as for the CertainLogic AND operator holds if $o_A = (1,1,1)$ then $o_A \wedge o_B = o_B$.

Security Levels

e_1 assigns security levels to classes of applications according to their value-at-stake (cf. [29] for a similar approach). A security level is a real number between 0 and 1. The higher the security level 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 relies on the risk profile of e_1, which is out of scope of this paper. General examples for security levels could be 0.99 for online banking, 0.9 for e-government applications, and 0.6 for social networks. Note, that as the trust validation requires the security level as input, the determination of the class of application is required. While an automated solution, for example, one based on content filtering (as also used to detect phishing sites [32]) or based on analyzing the type of entered data (cf. [22]) is desirable, this is out of scope of this work. In any case, the security level can be indicated by the entity, e.g., by using a visual slide control as part of the browser's user interface.

Validation Services

A certification path containing previously unknown CAs results in a low trust value. On the one hand this is intended, as it leads to the rejection of keys certified by unknown CAs. However, this is not necessarily due to malicious behavior, but due to the lack of information. Thus, whenever the key legitimacy is too low to consider a connection trustworthy, and the certainty is below a threshold rc which is set by e_1, validation services like notary servers (cf. Section 3.2) are queried to reconfirm a certificate. 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 user is asked for a decision.

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 sufficiently 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.

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 $p = (C_1, ..., C_n)$
- A trust view View
- An output of the trust validation R
- A list of new trust assessments TL
- 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: 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 $TA_i = (k_i, ca_i, S_i, o_{kl,i}, o_{it,i})$ to the corresponding trust assessments.
3. If $R = trusted$ then for $1 \leq i \leq n-1$ do
   (a) If $C_i \notin S_i$ add C_i to S_i
   (b) If $TA_i \in TL$ then add TA_i to View
   (c) If $TA_{i+1} \in TL$ then update $o_{it,i}$ with a positive experience.
4. If $R = untrusted$ then

```
    (a) Set $h = \{max(i) : TA_i \notin TL$ or the consensus $cons(vs_1(C_i), ..., vs_j(C_i))$
        $= trusted\}$.
    (b) For $1 \leq i \leq h - 1$ do
          i. If $C_i \notin S_i$ add $C_i$ to $S_i$
         ii. If $TA_i \in TL$ then add $TA_i$ to View
        iii. If $TA_{i+1} \in TL$ or $C_{i+1} \notin S_{i+1}$ then update $o_{it,i}$ with a positive
             experience.
    (c) If $TA_h \in TL$ then add $TA_h$ to View
    (d) If $C_{h+1}$ is not an untrusted certificate in View then update $o_{it,i}$ with a
        negative experience.
    (e) Add $C_{h+1}$ to View as untrusted certificate.
5. Return View
```

Example

An exemplary 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$_1$ → S-CA$_1$ → EE$_1$. The certificate of EE$_1$ is accepted. A positive experience is added to each involved CA.
(b) The chain R-CA$_1$ → S-CA$_2$ → EE$_2$ is obtained. The certificate of EE$_2$ is accepted. A positive experiences is added to each involved CA.
(c) The chain R-CA$_1$ → S-CA$_2$ → 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$_1$ → S-CA$_2$ was approved during prior observations, thus no negative experience is added to R-CA$_1$.
(d) The chain R-CA$_2$ → S-CA$_3$ → EE$_4$ is obtained. The certificate of EE$_4$ is rejected. Thus, the certificate R-CA$_2$ → 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$_1$ → S-CA$_2$ → S-CA$_3$ → 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$_1$ → 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. The question is, when enough evidence is available. One approach is to set the key legitimacy based on the number of positive experiences, or on the number of certificates contained in the trust assessment. To determine the best approach is due to future work.

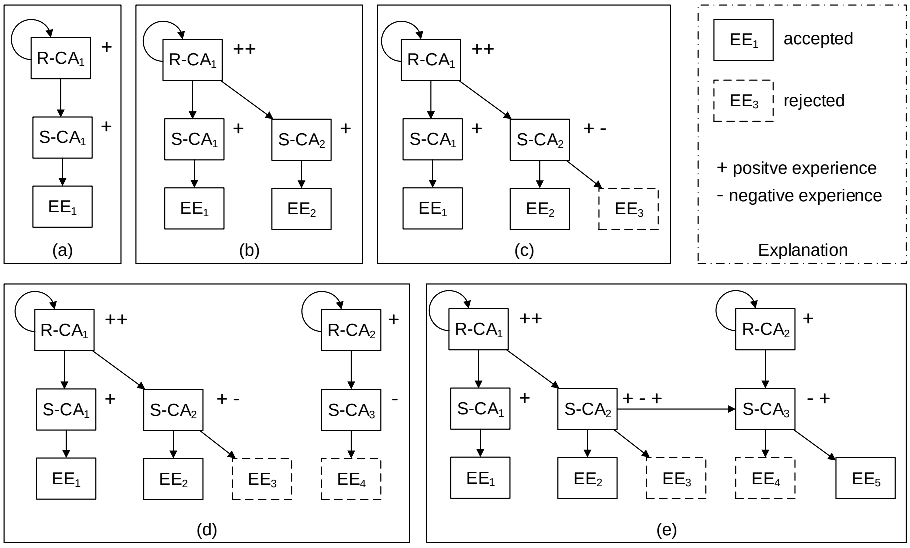

Fig. 1. Evolution of the trust view

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 Evaluation

To evaluate our concept, we first summarize the attacker model: First, we assume the user system not to be compromised in the sense, that an attacker cannot manipulate the trust view. Attacking user systems is out of scope of this paper. Second, we assume, that CAs themselves are in general not malicious on purpose. However, a CA's key can be used by an attacker to issue malicious certificates by compromising the CA's key, compelling the CA or by a CA failure, i.e. when the CA issues a certificate to an entity without properly checking the entity's identity. We further assume that once the malicious certificate is detected, countermeasures are taken like revocation and blacklisting of the certificate.

Attacking a CA does not end in itself but aims at attacking secure connections between users and web servers. In general the attacker either aims at a specific user group, i.e. he tries to eavesdrop, monitor or manipulate the communication of a specific group of people or the attacker aims at a specific service, i.e. he tries to attack the communication with a specific web server.

5.1 PKI Attacks

There are two cases, when a user obtains a certificate. Either, trust validation succeeds. In this case, the user solely relies on the CA system. Yet it implies, that the user has seen the certificate before, or the user has sufficiently many good experiences with the involved CAs. The requirements thereby increase with the security criticality of the application. Or, trust validation fails, in this case the validation services are queried to reconfirm the certificate. Given the certificate is trustworthy, the first case will occur most of the times (users in general stick to a limited set of CAs [1] as they repeatedly access the same services, and mostly web servers stick to the same CA when renewing their certificates). Thus, it is acceptable to have costly reconfirmation procedures as e.g. querying several independent notaries, and we may assume, that the validation services do not jointly provide false reconfirmations. This implies, that the attacker must compromise a CA which has a high reputation in order to be successful. However, this leads to several disadvantages for the attacker:

If he wants to attack a specific user group he must compromise a CA, with a high trust level in each of the user's trust views, otherwise the attack will be detected as the certificate is checked with the validation services by those users where trust validation fails. As the trust views are user-specific and not publicly visible, it is hard to identify such a CA trusted by all the users and it is questionable if the attacker can even manage to compromise that CA. Thus, the group of users where an attacker would be successful is reduced to a smaller group, which is unknown to the attacker and furthermore, the time span where a successful attack is possible is reduced due to the increased probability of being detected. The disadvantages for the attacker thereby grow with the number of the applications he tries to attack and their respective security levels. An attacker might also try to attack several CAs, but besides the increased difficulty to attack more than one CA at a time, he is not able to distinguish which user is to be attacked with which malicious certificate.

If the attacker aims at a specific service, he has the same problem of identifying a CA which is sufficiently trusted by all attacked users in order not to trigger the validation services and thus risking a fast detection. Besides that, users that used the service before have pinned the web server's certificate, which further increases the probability of a detection. The CA with which the attacker certainly has the highest success probability, is the CA that issued the certificates for the web server in the past. Again, it is in question, if the attacker can manage to compromise a specific CA.

In summary, this shows, that by the use of trust views an attacker can hardly employ accidental CA failures. The possible damage is reduced due to the limitation of the number of attackable users, while the attacker has a limited and unspecific choice of CAs. Furthermore, the damage a possible CA compromise may cause, highly depends on the CAs visibility in the certification business, which is a much more natural setting than each existing CA being equally critical. Furthermore, it becomes easily detectable which CAs need especially strong protection.

5.2 Attacks on Computational Trust

The trust views of the users govern when validation services are to be queried. Thus, attackers can try to attack the computational trust model in order to improve their success probability when employing a malicious certificate. The aim in this case is to disturb the correct functioning of the trust based decision processes. We discuss the standard attacks on computational trust, and how they apply to our system.

Whitewashing: This means, an entity (in our case a CA) re-appears in a system under a new identity to get rid of negative reputation. In our scenario this implies a CA that issued many incorrect certificates in the past. However, for a CA to re-appear in the system under a new identity and with a new key, there are significant hurdles. The CA's key needs to be certified by another CA which is already part of the Web PKI, or the CA needs to be incorporated into trusted root stores in order to pass path validation. Thus, whitewashing is prevented by standard mechanisms like audits or the required approval by a non-malicious CA, which a malicious CA is not likely to pass without essential changes in its processes and structure. This on the other hand can then justify such a whitewashing of the trust values. After re-appearing, before being trusted by users, the CA needs to demonstrate trustworthiness by correctly issuing certificates, which also requires to be chosen as a CA by web page operators.

Sybil attacks: A sybil attack means, that an attacker of a reputation system forges or controls other entities to produce many good ratings for a certain entity. Yet, such an attack has only limited relevance to our system as there are no unauthenticated entities that provide recommendations. An adversary could mount a sybil-like attack by attacking the underlying validation services in order to maliciously reconfirm certificates by a certain malicious CA and thus falsely improve the reputation of that CA. Yet, in this case, the attacker can directly exploit the malicious reconfirmation and therewith annul the trust evaluation. This shows the importance of the security of the underlying validation services. However, a successful attack requires both, the compromise of a CA and of the several validation services.

Exploiting slow trust adaptation: Such an attack means, that the good reputation of a CA can be exploited by an attacker, as it takes some time before the CA's good reputation is adapted when a sudden incident changes the CA's trustworthiness. Our system does not prevent such attacks. Thus, a CA that is used by many web pages and that built up a good reputation is a major goal for attackers. However, the possible loss of the good reputation and their public visibility provides strong incentives to such heavyweight CAs to put strong protection mechanisms in place.

In summary, for an attacker to benefit from attacks on the trust model, long term planning is required. Furthermore, additional mechanisms must engage which the attacker cannot influence or control, as for example web page operators must actually employ a CA in order that the CA becomes visible and trusted within the users' trust views.

5.3 Limitations

While trust views can significantly lower the risk of relying on a malicious CA, local 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 malicious certificate issued by such a CA. Yet, a CA compromise only threatens those entities, that trusted in the CA before the compromise, which limits the benefit for attackers. On the other hand it is also possible, that a connection is falsely evaluated not to be trustworthy, which relies 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 (which, for example, comprises the quality of the web page and its contents provided by the service provider). The latter is not addressed by the trust view concept and requires additional mechanisms like the Web of Trust[1] or commercial web page ratings[2]. However, such mechanisms require authentication, which is achieved via authentic public keys.

6 Conclusion & Future Work

We have presented the concept of 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.

Due to the user-specific reduction of the total number of trusted CAs, the possible attack surface for CA attacks that threaten the respective entity is reduced. Compromises and misbehavior of CAs that are not included in the trust view have no effect as such certificates always require additional reconfirmation and are untrusted when in doubt. CAs that have often been observed—and most probably will also often be observed in the future—achieve higher issuer trusts than CAs that are barely observed. This provides a trade-off between trusting in (a limited set of) CAs and the costly reconfirmation of certificates. Additionally, privacy problems are mitigated as validation services are only queried in rare cases and not for every connection establishment, which allows user profiling in the long run. Furthermore, the load for validation services such as notary servers is reduced. The application of certificate pinning, i.e. storing trusted certificates, further mitigates the threat of relying on malicious certificates for previously

[1] `https://www.mywot.com/`

[2] e.g., Norton SafeWeb `https://safeweb.norton.com/` or McAfee SiteAdvisor `http://www.siteadvisor.com/`

accessed services and prevents repeated reconfirmations. On the other hand, trust validation allows the automated exchange of stored certificates. Considering the user involvement, which is known to be problematic, the trust view concept also allows a fine grained steering: the user is only involved when the local knowledge and the data of the validation services is not sufficient for a decision. Thus, the total number of warnings is reduced and mainly limited to highly security sensitive services, which addresses the problem of warning fatigue.

Currently, we are working on the implementation of the presented concept. Challenges follow from the configuration adjustment to adapt an individual trust view to the needs of it's owner and to balance the system parameters. Thereby, the reduction of false decisions plays an important role. Furthermore, we will realize possible extensions. The model allows to combine local information with expert recommendations based on different indicators of trustworthiness or recommendations from other users. Challenges are the prevention of false or malicious recommendations and the authentication of the recommenders.

Besides that, a trust view allows to detect anomalies, like the unanticipated exchange of a locally stored certificate. We aim at making this local knowledge applicable for compromise detection.

References

1. J. Braun and G. Rynkowski. The potential of individualized trusted root stores: Minimizing the attack surface in the light of ca failures. Cryptology ePrint Archive, Report 2013/275, 2013. `http://eprint.iacr.org/`.
2. Carnegie Mellon University. Perspectives Project. `http://perspectives-project.org//`, visited July 2012.
3. D. W. Chadwick and A. Basden. Evaluating trust in a public key certification authority. *Computers & Security*, 20(7):592–611, 2001.
4. 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.
5. P. Eckersley and J. Burns. The (Decentralized) SSL Observatory. Invited talk at 20th USENIX Security Symposium, August 2011.
6. The EFF SSL Observatory. `https://www.eff.org/observatory`.
7. 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.
8. C. Evans, C. Palmer, and R. Sleevi. Public Key Pinning Extension for HTTP. Internet-Draft, 2013.
9. P. Gutmann. Pki: it's not dead, just resting. *Computer*, 35(8):41 – 49, aug 2002.
10. P. Gutmann. *Engineering Security*. 2013. Book draft available online at `http://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf`.
11. h online. Flame – oversights and expertise made for Windows Update worst case scenario. `http://h-online.com/-1614234`, visited July 2012.
12. h online. Fake Google certificate is the result of a hack. `http://h-online.com/-1333728`, visited Nov. 2011.
13. 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.

14. 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.
15. 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.
16. ICSI. The ICSI Certificate Notary, 2013. `http://notary.icsi.berkeley.edu/`.
17. A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9:279–311, 2001.
18. A. Jøsang and R. Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.
19. 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.
20. 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.
21. M. Marlinspike. Convergence. `http://convergence.io/`, visited July 2012.
22. 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.
23. 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.
24. PSYC. Certificate Patrol. `http://patrol.psyced.org/`.
25. 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.
26. 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.
27. 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.
28. J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying wolf: An empirical study of ssl warning effectiveness. 2009. available online at `http://static.usenix.org/event/sec09/tech/full_papers/sunshine.pdf`.
29. 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.
30. 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.
31. 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.
32. 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.
33. P. R. Zimmermann. *The official PGP user's guide.* MIT Press, Cambridge, MA, USA, 1995.