Reputation Systems for Trust Management in the Web PKI

Master-Thesis by Jiska Classen from Kassel

March 31, 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
CDC

Reputation Systems for Trust Management in the Web PKI

Vorgelegte Master-Thesis von Jiska Classen aus Kassel

1. Gutachten: Johannes Buchmann
2. Gutachten: Johannes Braun

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 31. März 2014

(J. Classen)

Abstract

This thesis' goal is to reduce the attack surface of the existing Web Public Key Infrastructure (PKI) by applying user trust as in real world scenarios of human trust establishment. Reputation systems can be used to exchange Certification Authority (CA) trust information and lower the risk on relying on malicious CAs.

The matter in question is connection trust, determining whether a SSL/TLS connection between a client and a server is secure because all involved certificates were issued by trustworthy CAs. A secure connection prevents attackers from reading and manipulating data during a transmission over this connection. Connection trust is no assurance for the trustworthiness of a server, for example the server could run an online shop selling poor quality products.

In this thesis, it is discussed how existing reputation system approaches fit into the current Web PKI scenario or how they have to be adapted. The reputation system has to be secure against attacks, since it is an attack vector for connection trust as CAs themselves. Therefore an analysis framework fitting CA reputation system attacks is proposed in this thesis. Within the evaluation, all presented reputation system approaches are compared and evaluated by the attack analysis framework.

Both, centralized and distributed CA reputation systems, are discussed. More details about the centralized solution called CA-TMS can be found in *"CA Trust Management for the Web PKI"*[4], of which portions of this thesis consist and which contribution was part of this thesis work.

Contents

1 Trust in the Web PKI

1.1 The Web PKI

The current Web PKI is based on X.509 certificates[7]. Common protocols like SSL/TLS relate on these certificates, leading to a wide usage. If a client connects to a web service, it presents its certificate and certification path during a handshake. The client checks if all certificates within the certification path are valid and if it starts with a trusted Root CA.

X.509 certificates confirm the binding between a subject's identity and its public key. They contain the Distinguished Names (DNs) of an `issuer`, the CA, issuing and signing the certificate and DN of the `subject`, an entity associated with the public key stored in the `issuer public key` field.

The Web PKI's structure is hierarchical. The root of trust is the root store containing so called Root CAs, whose certificates are typically shipped within software like web browsers. Root CAs issue certificates for Sub CAs, which can issue certificates for end entities or their Sub CAs and so forth. The `pathLenConstraint` limits the hierarchy's depth. Therefore, some Sub CAs can only issue certificates for end entities. Furthermore, CAs can cross-certificate each other leading to an even more complex certificate structure.

A binding between a Root CA and an end entity can include multiple Sub CAs, called certification path. During *path validation*, the certification path is checked for correctness and validity.

1.2 Current Problems with the Web PKI

Secure Sockets Layer (SSL) was designed by Netscape in 1994. Compared to now, there were only a few websites with a need for certificates and secure transmission – and no applications like online banking or e-commerce. Having a few CAs to validate and issue certificates seemed realistic.

20 years later, many web services support Transport Layer Security (TLS), the successor of SSL. The web grew from roughly 1000 pages in the beginning of 1994 to at least 2.14 billion pages in 2014[30][8]. An internet wide scan from 2012 shows that 7.7 million HTTPS hosts rely on more than 5.8 million distinct X.509 certificates[14][23][p. 5]. SSL/TLS is not only used to secure websites but also for sending emails, file transfer, chat protocols and many more. The standard is so widely adopted, that making changes is hard and backward compatibility is required.

The average user has no idea, who these certificate issuing CAs are but somehow needs to trust them. To simplify this trust issue, web browsers are shipped with a huge list of Root CAs. Since many CAs can issue certificates for Sub CAs, users transitively trust a tremendous amount of CAs not even contained in these lists due to the process of path validation. Starting with the 317 CAs directly trusted by Mozilla, Apple, Microsoft and OpenSSL and according to a current HTTPS web scan this means, that an user trusts 1207 Root and Sub CAs by default[23][p. 5]. Depending on the root store size and the scanned services, this number is even larger.

Relying on many CAs leads to a large *attack surface*. There is a high risk that one of them gets attacked or creates false certificates on purpose. In the consequence of the large root stores, every nation can issue valid certificates for anybody. Having a single malicious CA within the root store suffices for an attack. Due to the huge amount of certificates required for the Web PKI, reducing the number of CAs issuing certificates is no realizable approach.

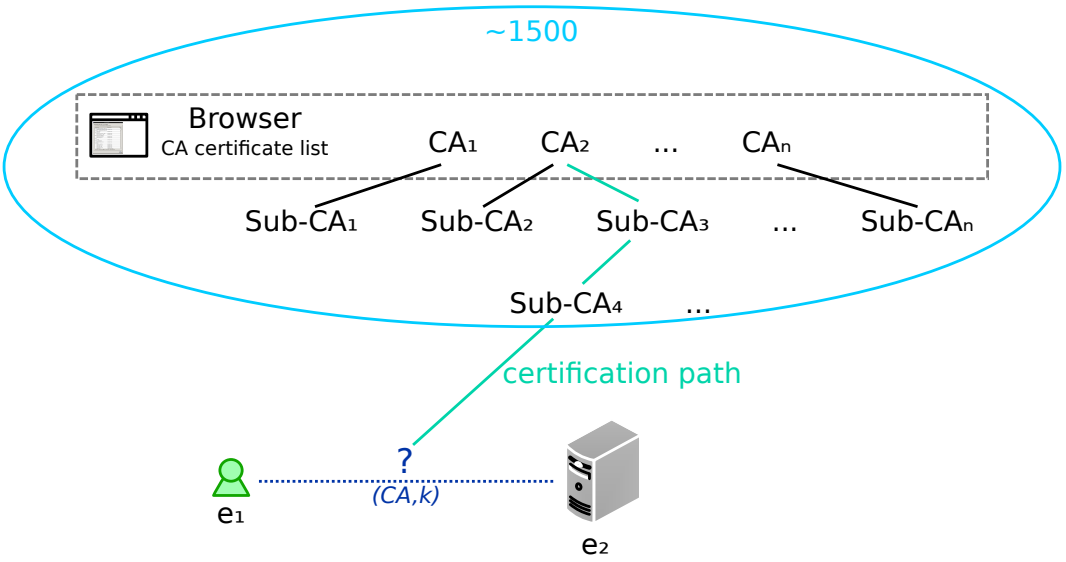

Figure 1.1: Trust in the current Web PKI

Once a new CA was introduced and adopted to common lists, users will trust it forever and context independent. Even though X.509 certificates can be revoked or removed from those lists, users trust in entities like popular browser vendors and wide spread CAs leading to a long durable trust model. Of course users could remove certain CAs from their lists or even try maintaining them by hand – probably leading to many certificates considered as invalid. And how should the average internet user be able to decide whom to trust in that many cases?

There has been a tremendous amount of CA incidents in the past. For example DigiNotar, a Dutch CA, was hacked and issued certificates including `google.com`, `skype.com` and `microsoft.com` which were used for Man in the Middle (MITM) attacks on Gmail users in Iran[13]. After this incident, DigiNotar was liquidated. Comodo, a very large CA, had an attack on one of their reseller account[5]. They revoked the fraudulent certificates and nothing happened to them. Removing Comodo from root stores would have a large impact because its within the top 10 issuers, according to [17][p. 12]. Despite from incidents, there are CAs like StartSSL[1], signing certificates for free, that are contained in popular browser's root stores.

Therefore, the main questions for new approaches in the Web PKI are: Trust whom? For how long? In which context? The current Web PKI means trusting a prescribed set of people forever.

A solution should meet the "trust agility"[27] criteria:

1) *a trust decision can be easily revised at any time*

2) *individual users can decide where to anchor their trust*

1.3 Trust Views

The user local trust concept in this work is based on the paper *"Trust Views for the Web PKI"*[3]. It already meets the trust agility criteria since it enables users making their own trust decisions for exactly those CAs they need.

There are many CAs contained in root stores, but most users only require a small subset of these to validate certificates successful. The numbers in a recent study based on web browser histories show that limiting the number of trusted CAs depending on the individual user requirements has a high potential[2][p. 9]. Furthermore, there are overlaps depending on certain user

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

groups. Reducing the set of CAs to those required by a specific user would significantly reduce the attack surface.

Based on these findings, the concept of trust views was developed[3]. Trust views limit the set of trusted CAs and enable users making their own trust decisions. Instead of just having the binary model of valid and invalid CAs within their root store, meaning to either fully trust them or not to trust them at all, users get variable issuer trust recommendation. Depending on the security level required by a site's context (e.g. maximum security for online banking), trust decisions for a connection are made.

Basically, a trust view contains experiences made by an user while browsing websites. Experiences can be represented in an experience space, opinions about issuer trust can be represented in an opinion space. Based on existing local experiences and using the *CertainTrust* model[36], the opinion space is used to combine experiences made with different CAs.

Figure 1.2 shows an example trust view containing positive and negative experiences. According to evaluated browsing histories, trust views reduce the attack surface by at least 95%, even when using the minimal security level for all websites.

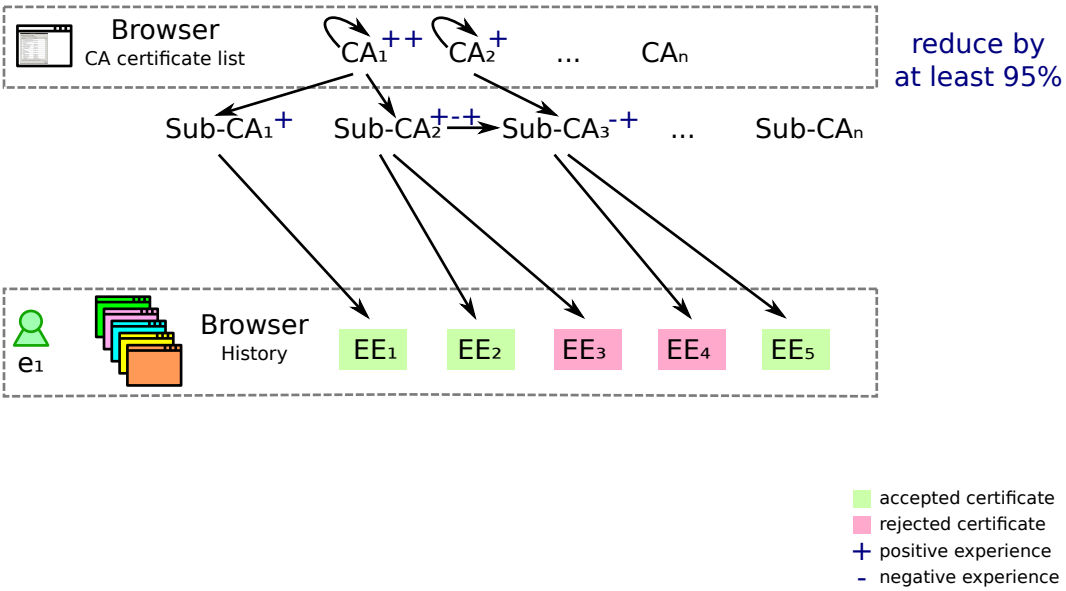

Figure 1.2: Trust view example

Trust validation includes the standard path validation, but is also depending on an application specific security level and a required certainty for the validation result. Based on the output of the trust validation, the local trust view is updated by collecting new and either positive or negative experiences are for the involved trust assessments.

A drawback of this approach is that it requires quite a long time to stabilize. According to [2], it takes several months until a user has seen almost all required CAs, depending on his browsing behavior. During this time, the user would see many unknown CAs and hence have certificates with an unknown issuer trust. Since the average internet user has no idea when seeing a certificate if it is valid or not, *external validation services* have to be queried. Only in case these services don't know if the corresponding experience should be positive or negative, user interaction is required.

Figure 1.3 shows the process of trust validation and bootstrapping a local trust view until trust decisions can be made autonomous without querying any external services. The time until this system works autonomous varies a lot. In case a completely unknown CA with no direct or indirect experiences is seen later, external services have to be used again.

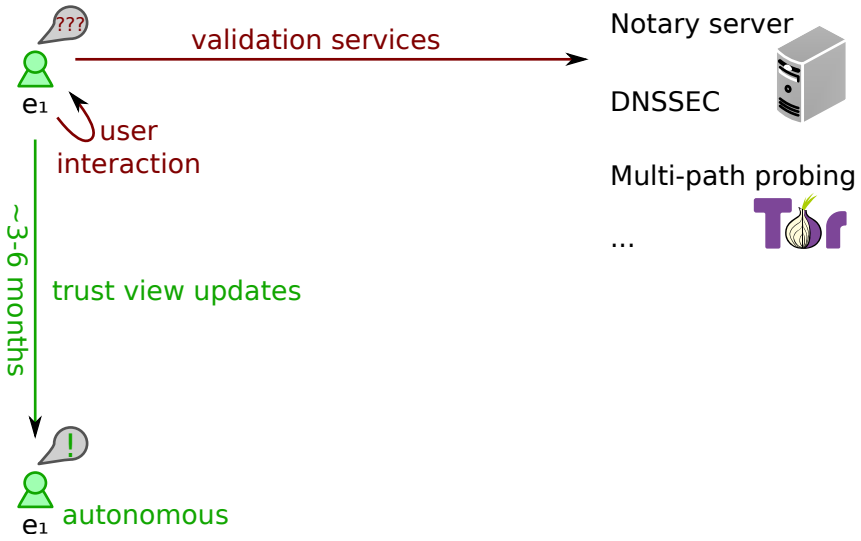

Figure 1.3: Trust validation and bootstrapping

To speed up the bootstrapping process and make the system work autonomous, one can initialize the trust view with all certificates that belong to web pages contained in the browsing history – requiring the user does not clear its history periodically due to privacy reasons. During the bootstrapping process, no MITM should be present and none of the CAs should be compromised. To ensure this, one can initially use again external validation services to receive more reliable information but to the cost of privacy. Furthermore, external validation services are still centralized trust anchors that don't meet user specific trust demands.

Details about trust views can be found in [3]. Important concepts and additional details of trust views are briefly introduced when required for a basic understanding of what a CA reputation system does and why.

1.4 Adding a reputation system

Adding a reputation system enables users publishing their trust views containing opinions about CAs. Using external opinions as issuer trust recommendations tremendously reduces the bootstrapping time.

When taking other opinions into account, user specific trust requirements need to be preserved. The study in [2] showed, that there are different CA usage patterns depending on ethnic background. A reputation system containing many trust views can return user specific results, e.g. by comparing trust view similarity and only taking opinions from nearby trust views into account.

Publishing trust view information is a *privacy* tradeoff. Sharing information helps calculating authentic trust values but goes to the cost of privacy. Improving privacy despite sharing some information will be discussed in this paper.

An approach for adding and using a centralized reputation system for trust views was already discussed in the paper *"CA Trust Management for the Web PKI"*[4] which contribution was part of this thesis' work. Additional approaches including distributed ones and improvements will be discussed in detail in this thesis.

1.5 Related work

There are many proposals on how to improve the existing Web PKI. This section briefly discusses the most interesting and important ideas. Chapter 3 gives an overview and comparison of existing reputation system approaches, therefore they will not be discussed here.

Collecting local information

Certificate pinning is a technique saving a server's certificates when visiting the server for the first time and afterward using this certificate for a comparison if this server is visited again. In case the certificate changed, a MITM attack might be ongoing. With Certificate pinning, trust is established on the first use. SSH host key lists and browsers requiring user interaction when a host key or self-signed certificate is seen for the first time follow the same principle, called Trust on first use (Tofu)[37].
This approach has multiple problems. First, there are problems if the attacker is present during the first connection attempt. Second, certificates might change or get updated regularly, leading to a legitimated certificate change that would be detected as attack.

The more sophisticated approach of using *trust views* to locally compute new trust values has already been discussed.

Notary servers and multi-path probing

Notary servers are instances a client can show a certificate to get a response if it is valid or not. A notary has the possibility to keep track of a server's certificate over time and to do multi-path probing. Notaries increase trust – but to the cost of privacy, since users need to contact a notary for each SSL/TLS connection.

The idea behind *multi-path probing* is to check if the same certificate is shown via multiple routes to the same server. This prevents from many cases where a MITM is present like manipulated wireless networks or networks near to the client. It does not help in case first-hop-routers from the server or DNS servers are compromised.

Perspectives is using sets of multiple notary servers keeping track of certificate changes of many services[37]. It helps making more reliable trust decisions by providing spatial and temporal redundancy. Users can query multiple sets of notary servers to check results.

DoubleCheck uses multi-path probing without notary server[1]. It is designed to check self-signed certificates which can not be checked due to the missing trust anchor to one of the Web PKI's Root CAs. There is a ready to use firefox plugin[2].
It does so by using The Oninon Router (Tor), a anonymizer establishing virtual tunnels through a network of participating Tor nodes. Establishing multiple connections via Tor will result in different paths to the same server. In case the attacker is not close to the server, he is not present in all these paths and will be detected while doing a MITM attack.
Compared to *Perspectives*, *DoubleCheck* does not require new infrastructure like expensive notary servers. Since no CA was present when signing the certificate, the identity of the person creating the certificate is not checked and therefore it doesn't help against fake servers controlled by an attacker, e.g. for a phishing attack.

[2] http://www.cs.columbia.edu/~mansoor/doublecheck/

Based on *Perspectives*, the firefox plugin *Convergence*[3] was developed[27]. *Convergence* does not use Tor to improve anonymity (as *Doublecheck* does) but by using notaries as bounce to other notaries. The notary an user is contacting directly only knows who the user is but not which web pages and certificates he asked for, the other notaries know the web pages the user asked for but not who the user is. To reveal an user's browsing history, multiple notaries need to collude. Privacy is improved by caching. Furthermore, *Convergence* notaries can implement different mechanisms than multi-path probing like DNSSEC, BGP data or just normal path validation. An user can then set the option that multiple notaries have to agree with their results.

Certificate transparency

Certificate transparency means that X.509 certificates get publicly logged as they are issued or observed[26]. Users can choose not to trust certificates not contained in any log. Logs get published via servers allowing to do queries and submit certificate changes. Everyone could audit if certificates were issued correctly.

CAs might refuse adapting certificate transparency, because maintaining these logs increases the effort for issuing certificates.

Restrict the CAs' signing scope

CAge is an approach based on web scans to reduce the overall attack surface of the Web PKI[23]. Current web scans showed that most CAs only sign a few Top Level Domains (TLDs) but they do not use the X.509 certificate field `NameConstraints` to restrict their scope to these TLDs. Therefore, *CAge* introduces a separate list based on web scans for additional checking if a CA is usually signing the TLD of the certificate in question resulting in a 75% decrease in attack surface. Adding a threshold requiring a CA already singed multiple certificate of the same TLD reduces the attack surface to 11.1% of the original.

In contrast to adding certificate transparency, no infrastructure and organizational changes are required. TLD lists can be distributed as browser plugin and have to be updated regularly to reduce false detections.

Binding certificates to DNS

DNSSEC is a security extension to Domain Name System (DNS) mitigating unauthorized modifications like DNS spoofing[24]. It can be combined with DNS-Based Authentication of Named Entities (DANE), a standard for publishing X.509 certificates or authorizing specific CAs for signing within the DNS record type `TLSA`[16].

Zone administrators are responsible for doing changes within *DNSSEC*. For the `.com` root zone, VeriSign is responsible – but they also are a CA issuing certificates for `.com` domains. In a case like this, *DNSSEC* helps against MITM attacks but the root of trust remains. Furthermore, DNS records need some time until they reach a client due to caching mechanisms and intermediate DNS servers.

What is missing?

Many interesting ideas and solutions are among the introduced techniques. All of them could determine if an experience is positive or negative to a certain extent. Some of them could be even used to improve the existing trust view approach or could be considered within the additional reputation system.

Even though some of them enable users making their own decisions and therefore meet the trust agility criteria, none of them enable users trusting single persons or groups that are similar

[3] http://convergence.io/

to them. They do not distinguish aspects like the nationality of users which could result in different trust decisions for the same certificate.

Some approaches enable users making their own decisions, but they remain local. Adding a reputation system would enable users communicating about their trust decisions.

Most solutions are potentially managed by large companies like notary servers requiring a lot of resources or central TLD list distribution. Therefore, the roots of trust have many intersections with existing CA root stores. Instead, users require personalized trust anchors.

2 Attacker Model

In the following, an attacker model for reputation systems is developed. Based on this, CA reputation system approaches will be evaluated in chapter 6.

2.1 Reputation System Basics

Trust decisions in the later on proposed trust management system are guided by a reputation system collecting feedback about CAs from users. Instead of attacking CAs directly, an adversary could attack the reputation system in order to manipulate trust into them. This means that the underlying reputation system needs to be secured against attacks.

In a reputation system, users help other users to decide whom to trust. Users of the reputation system don't know each other, they are strangers from the internet. Thus, mechanisms to ensure these strangers' trustworthiness are required.

Reputation systems can be classified as *soft security*, they are a social control mechanism[19]. In soft security, no hard distinction between good and bad is possible. The same action might be unethical or not, depending on context, point of view and other factors – there is no generally accepted policy for good behavior. Security policies are implicit and collaboratively emerged by a community.

There are many popular implementations of reputation systems like eBay's feedback forum, Amazon's review rating or Slashdot's moderation.

2.2 Use Cases

Attacker goals

MITM: An attacker wants to inject a malicious certificate with a known private key in order to do a successful MITM attack on a SSL/TLS connection. To archive that the malicious certificate appears trustworthy he can try manipulating the reputation system.

Since the reputation system was introduced to improve the Web PKI's security, it is assumed that the main goal of an attacker is making an untrusted CA appear to be trusted within SSL/TLS connections.

Economic: A CA with better ratings than others might be able selling more certificates and thus will have an advantage in competition.

Privacy: Reading trust view data from the reputation system could expose privacy critical information.

Components under attack

An attacker aims at manipulating the reputation system, an user's system with its local trust view or the communication between them.

Vulnerabilities on software different from CA trust management on the reputation system or an user's system are out of scope. In case an user's local system would be manipulated by an attacker to disseminate fake ratings, the scenario would be equivalent to an attacker creating a new account uploading such ratings.

Communication between user systems and reputation system as well as between multiple interacting reputation systems has to be secured.

The reputation server itself is not manipulated, e.g. the attacker has no access to its trust view database and cannot change algorithms executed on the server.

Hence, an attacker can only indirectly modify an user's local trust view and trust views within the reputation system. If the reputation system has no or weak defense mechanisms, there are still a lot of possibilities to do so.

Attacker capabilities

Attackers have the possibility to observe and manipulate *network traffic* between users and other entities like the reputation server and visited SSL/TLS services.

It is assumed that the attacker is *under control of one CA* or Sub-CA within the user's root store when performing a MITM attack. Due to this, the maliciously issued certificate will pass standard path validation. (Trust views aim at reducing the attack surface for this scenario by distrusting CAs within the root store until sufficient experiences showed them to be trustworthy as shown in section 2.3.3.)

Attackers are able to actively interact with the reputation system as normal *users* which means registering accounts, uploading trust views and requesting issuer trust recommendations. They have no direct access to the reputation server's database nor the users' systems.

Attacks are not random actions of single users. It is assumed that *actions depend* on previous actions to archive certain goals. Furthermore, multiple attackers can *cooperate*.

Attacks can have limited time or a *long duration*. Being able to observe and manipulate network traffic between an user and the reputation server for a long period of time can be possible. An attacker could introduce very small changes in many trust views on the reputation system over a long period of time and maybe remain undetected, but such an attack requires a lot of planning and resources. Limiting an attacker's time gives him less capabilities and raises the risk of being statistically detected, e.g. with methods proposed by Yang[38].

2.3 Analysis Framework

2.3.1 Reputation System Components

According to [33][p. 2], reputation systems collect, distribute and aggregate feedback. In [15][p. 4] reputation systems were split into the same components to build an analysis framework, but they were given the names formulation, calculation and dissemination.

Understanding a reputation system's components helps understanding attacks from the next section. Figure 2.1 gives an overview of how components interact with each other and with users.

Alternative classifications in components exist, like information gathering, reputation scoring and ranking, and taking action for P2P reputation systems in [28] with the last component at a focus on how to motivate peers to contribute and how to punish adversaries. Anyhow, the taking action component is actually a defense mechanism. A more fine-grained list of characteristics instead of main components can be found in [25][p.5].

Collection

First, a reputation system collects user opinions formed by experiences made during the browsing process. This step has to be adapted from the existing trust view approach and needs to be integrated into the reputation system.

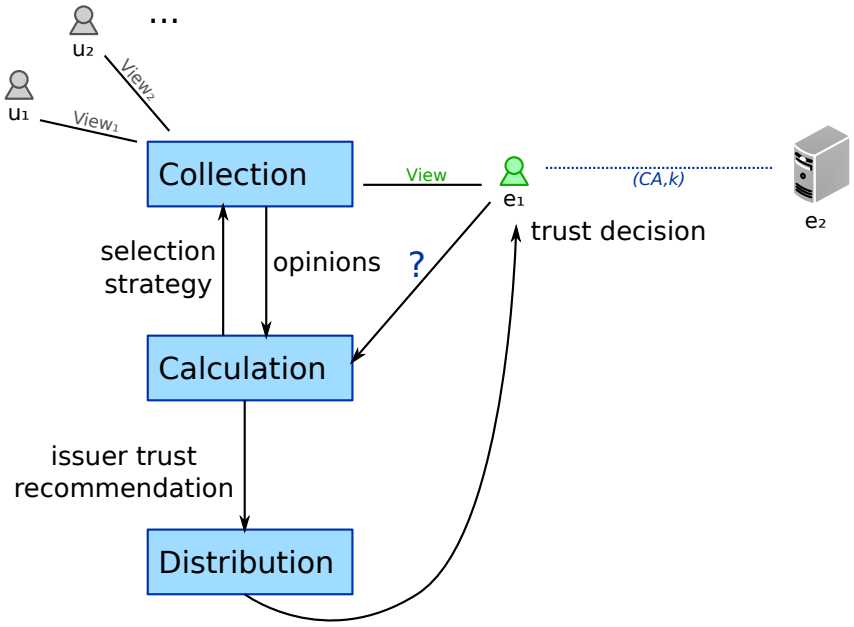

Figure 2.1: Reputation system components

A single *experience* resembles if a certificate was issued correctly or not. Experiences are binary: either positive or negative. If an experience is positive, the SSL/TLS connection is trustworthy. Based on multiple experiences about issued certificates, users locally compute an *opinion* using the CertainTrust model by Ries[36]. An opinion o_{it}^{ca} represents the trust of an user in a CA issuing trustworthy certificates for CAs, while an opinion o_{it}^{ee} represents the trust of an user in a CA issuing trustworthy certificates for end entities. Each opinion consists of a triple (t, c, f). The value $t \in [0; 1]$ is the trust in the correctness of the statement, here that an issuer signs certificates correctly. The certainty $c \in [0; 1]$ represents the probability that t is a correct approximation and 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.

Aggregation and Calculation

Based on user opinions, the reputation system calculates reputation scores, in our case, an *issuer trust recommendation*. It is important whose opinions are considered when calculating a recommendation for a specific user. Therefore, a resistant *selection strategy* has to be implemented, e.g. based on trust view similarity.

Distribution

Issuer trust recommendations have to be disseminated to users helping them making trust decisions. Attackers should not be able to block or manipulate this information.

2.3.2 Attacks

Even though the introduced components help to get a basic understanding of what could be attacked, many attacks harm multiple components. Therefore, building and categorizing an analysis framework on how entities misbehave during an attack as in Koutrouli[25] is more reasonable. General defense mechanisms are out of scope of this work. However, one can check

if a reputation system is resistant against known attacks. This helps understanding weaknesses and do further improvements.

The attack classification from this section will be used to evaluate the proposed reputation system for CA trust management. It is based on Koutrouli[25] which categorizes attacks on P2P reputation systems, but is more general for centralized and distributed approaches. Other related work like Hoffman[15][p. 12ff] and Josang[21][p. 4] only lists known attacks but doesn't make a categorization by attacker goals. Some attacks are known under multiple names.

*All attacks marked with **(*)** do **not apply** to the CA trust view scenario.* Some attacks can be excluded in advance independent from a CA reputation system's implementation details, e.g. because users share opinions about CAs, but CAs don't share opinions about users. In many other reputation systems, participants can have the role of both, user or service provider.

Figure 2.2 gives an overview of all relevant attacks.

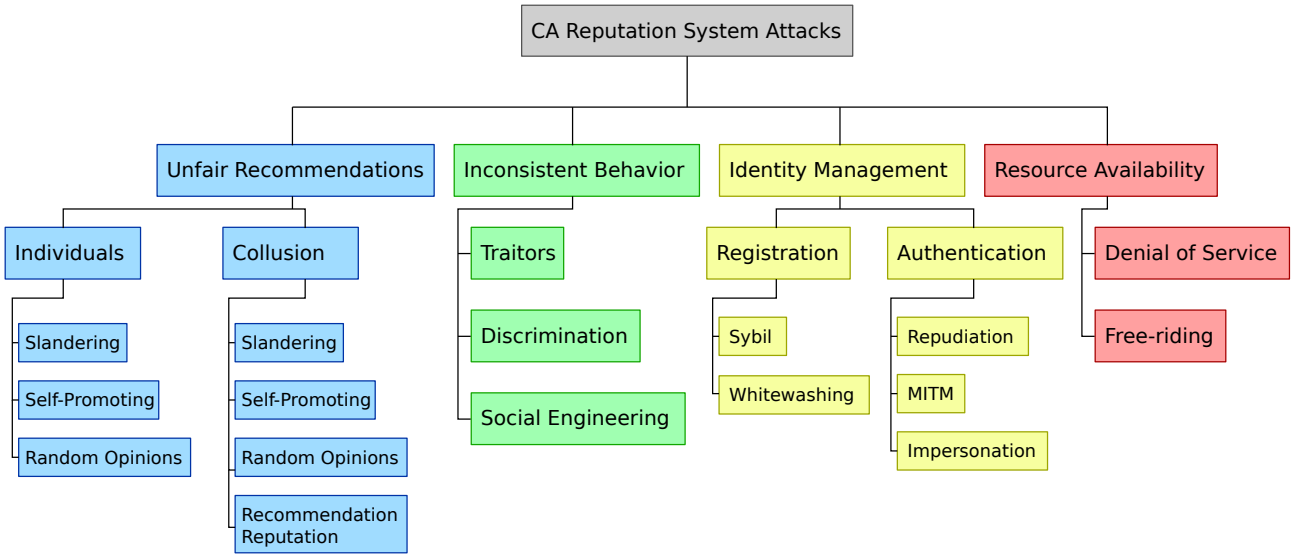

Figure 2.2: CA reputation system attacks

Unfair recommendations

An attacker could introduce experiences or opinions not resembling reality.

Unfair recommendations from individuals

Slandering / Bad mouthing / Discrimination: Slandering aims at lowering the reputation of a specific CA.

As the attacker does not directly benefit from decreased trust in CAs, but he might aim at disturbing the proper functioning of the reputation system. When asked for an opinion about CAs, an attacker could discriminate only a specific CA. In the consequence, other ratings from this attacker would look accurate and the discrimination might be given a higher weight.

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 into an user's local trust view.

Self-promoting: Self-promoting are actions of an attacker making him or a CA under his control appear more trustworthy. This is exactly the main attacker goal since it increases the probability that an attackers malicious issued certificate will get accepted when establishing a secured connection.

A self-promoting attacker can approach his goal either by injecting manipulated certification paths containing positive experiences or by a Sybil attack to add more positive opinions to the reputation system itself. Since the trust view approach distinguishes between o_{it}^{ca} and o_{it}^{ee}, an attacker is not able to increase its reputation concerning end entities by signing certificates of other CAs having a positive reputation.

Unfair praises (*): In a reputation system like eBay, users tend to give unfair positive feedback for negative transactions, since they fear of receiving unfair negative feedback in return. This results in overall high feedback scores even if there were negative transactions.

In the CA scenario, where users publish opinions about CAs, but CAs have no possibility to disadvantage users, this is not the case. An user has not to fear any consequences when he has a negative opinion about a CA.

Inaccurate recommendations (*): Some opinions of an user might have a high uncertainty and therefore lead to inaccurate recommendations. This is not the case in our scenario, since the CertainTrust based opinions in the trust view approach include a level of certainty.

Random opinions: An attacker could create a massive amount of random opinions within its trust view. He could so e.g. by crawling the web by random and creating a trust view according to this. This helps the attacker in case the reputation system prefers users with larger trust views or awards them for publishing opinions. However, a good selection strategy might filter out such random trust views.

Proliferation (*): In case a user has to choose between multiple service providers, an attacker has a higher probability to be chosen by the user if he offers the same service over multiple channels.

In the CA scenario, there is only one service provider (the CA who issued the certificate in question) which is either trusted or not. Users cannot choose between multiple certificates for one website.

Collusion / Orchestrated

In a collusion attack, multiple malicious users work together to have a stronger influence on recommendation results. The same result could be reached if the attacker somehow exploits the identity management system (typically with a Sybil attack) or has systems of multiple users under control.

Collusive slandering or self-promoting: If multiple users spread false opinions about a certain CA, this could tremendously change its reputation. This could even trick the trust view selection strategy and include such false opinions in many issuer trust recommendation calculations.

Collusive deceit (*): In this attack, a group of users performs bad transactions but rate each other positive. Since CAs don't rate each other, this type of attack does not apply.

Collusive reducing recommendation reputation: In case trustworthiness of users is determined by the number of equal opinions, an attacker could create conflicting opinions to decrease the recommendation reputation of honest users. The selection strategy should somehow handle such conflicting opinions.

Inconsistent behavior

Traitors / Playbooks

A malicious CA could issue many trustworthy certificates to gain a positive reputation and then change its behavior issuing malicious certificates. Even multiple oscillating behavior changes from good to bad are possible. This might hinder the reputation system from updating the CA's reputation with an adequate value. Additionally, if the reputation system reacts slowly to

changes, an attacker can exploit the time period caused by *reputation lag* until its behavior leads to consequences.

A special case of this attack is *value imbalance exploitation (*)* where positive reputation is earned with cheap transactions (e.g. selling many cheap items) to push an expensive transaction (e.g. selling one high-priced item). The trust view approach distinguishes between o_{it}^{ca} and o_{it}^{ee} which hinders a CA from getting positive experiences by signing certificates of existing, well-behaving CAs to gain a positive experience and then maliciously signing certificates for end entities. Within the class of end entity certificate opinions and the class of CA certificate opinions, each certificate experience is considered equal and therefore this attack does not apply.

Another special case is the *exit (*)* strategy, where a CA doesn't fear loosing a good reputation because it knows that it will shut down its services within a short time anyway. One can assume that this case is rare since building up a CA requires a lot of effort.

Honest behavior but no recommendations (*)

Users with a high reputation might not want to share good opinions about other users because this would increase the reputation of competing users. In the CA scenario, where users rate CAs but CAs do not rate other CAs, this is not the case.

Discrimination when providing services

A CA could behave good when issuing certificates in most cases but behave bad when issuing certificates for a very small subset of users, e.g. only when issuing a single certificate for a small website. In this case, most users make a positive experience with this CA and a victim seeing this certificate for the first time might get a positive issuer trust recommendation.

Social Engineering

Social Engineering means an attacker tries to manipulate users such that they behave different. Various attacks are possible, for example:

1. An attacker could try to influence if experiences were positive or negative. Each experience resembles if a certificate the CA issued was considered trustworthy or not. This changes the overall opinion about a CA. However, the case an users behavior could be manipulated considering experiences is rare, since he is only prompted if its trust view has not sufficient information.

2. An attacker could try to make the user change his local settings, e.g. setting the security level l to a low value.

3. Furthermore, an attacker could try to gain information about the user's browsing behavior, for example by talking to the user to find out some of his interests, and then upload manipulated trust views that are very similar to this specific user to increase its trustworthiness.

Identity management related

Registration policy related

There are multiple challenges regarding the registration policy within a CA reputation system:

- Bootstrapping a new CA is very costly. CAs don't have to register at the reputation system but are contained in trust views.

- Users should be able to register at the reputation system for free. The registration process should not be automated easily by an attacker. However, respecting an user's privacy, no personal information should be included in the registration process.

- Users want to remain anonymous but a somehow unique identity is required to defend against attacks.

Sybil attack: Performing a Sybil attack means the attacker forges or controls a large amount of entities and acts on behalf of them. When targeting the reputation system, in many cases 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.

Whenever an attacker is able to register itself with several forged identities at the reputation system (or the attacker controls the systems of already registered users), 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 is controlling a sufficient large fraction of a reputation system's user base, the attacker effectively controls the content of the recommendations generated by the reputation system.

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

CA Whitewashing (*): 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 the CA scenario. 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.

User Whitewashing: An user could enter the system under a new identity if other users decided its opinions not to be trustworthy. Depending on how trustworthiness is determined, this helps him disseminating its opinions again until its behavior is punished. Therefore, a sufficient *initial window* policy is required.

Authentication policy related

In case there is no way to proof an entity's identity, the following attacks can be possible. This is no issue for the CA itself since only a CA is able to issue certificates under its name. Anyway, proving the identity of an user might be a problem and therefore be an attack vector.

Repudiation – false accusation: If the reputation system is not able to verify for which actions an user was responsible, an attacker would be free to do the following things without being detected:

1. issue unfair or fake opinions

2. refuse sending of opinions

3. accuse other users for misbehaving

MITM: Within connections of the reputation system, a MITM could be present. In a centralized reputation system, the MITM could try to read and modify uploaded opinions and downloaded issuer trust recommendations. In case of a distributed reputation system, where users might

forward queries to other users and so on, the impact of a single MITM might be even worse since it is easier affecting more users.

A MITM could *miscommunicate information*, e.g. omit or modify it. Furthermore, a MITM could try to gain information about an user's privacy and therefore *breach privacy*.

Impersonation: An attacker could steal an user's identity and trust view, e.g. in order to successfully spread unfair opinions in the name of an user that was considered to be trustworthy.

Resource availability

Denial of service

An attacker could try to overload the reputation system by uploading a large number of random opinions. This could be an issue in both, centralized and distributed approaches, either when the central system cannot process and save new opinions fast enough or if the distributed system forwards opinions and overloads the network.

Furthermore, an attacker could introduce a delay when asked for opinions to slow down the issuer trust recommendation calculation or don't answer at all. In the consequence, the user has to wait a long time until he gets help on his trust decision which is not only a reputation system performance issue but also an usability issue.

Free-riding

An user might want to profit from opinions of other users but refuse to share its own opinions. Even honest users might do so, e.g. because of fearing privacy leakage.

2.3.3 Attack surface analysis of trust views

In [23][p. 10], a simple attack surface analysis formula is proposed. It is attack independent since it only measures how many domains CAs in the Web PKI (denoted *CAs*) can sign if their scope is reduced to certain domains or TLDs ($dom[ca]$). The resulting formula for the attack surface is $AS = \sum_{ca \in CAs} dom[ca]$.

This attack surface formula can be adopted to the trust view approach as shown in [4]:

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

with

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

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

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

The attack surface reduction depends on an user's individual trust view View. Domains are distinguished by their required security level, e.g. the maximum security level dom_{max} contains applications like online banking and within this domain the maximum security level l_{max} is required. Security level values for l_{max}, l_{med} and l_{min} are user specific settings.

As [4] shows, even with a very low security level applied for all domains, the attack surface is reduced by more than 95% if the trust view approach is working correctly. Raising the security

level to l_{max} for all sites even reduced the attack surface to at least 98.2% in all examined trust views. Therefore, building a reputation system based on the trust view approach is tremendously increasing security.

3 Reputation Systems

Section 2.1 already gave a basic idea of goals and a basic functioning of a reputation system for trust management in the Web PKI. This chapter gives further details about reputation systems in general and ways to implement them.

In the basic scenario there is an entity e_1 with trust view View and another entity e_2. e_1 establishes a SSL/TLS connection with e_2 and needs to decide whether the connection is trustworthy. First, e_1 tries to derive all necessary information locally. In case trust validation fails due to missing information, external validation services like notary servers, DNSSEC and other solutions from section 1.5 can be queried. External validation services only help deciding if single experiences were positive or negative, but they don't recommend an opinion about issuer trust. Therefore, bootstrapping requires much time until sufficient experiences were collected. This can be speed up by a reputation system recommending issuer trust values.

3.1 Disambiguation

Recommender Systems

Recommender systems use Collaborative Filtering (CF) to take ratings depending on a subject's taste as input[19][p. 221]. They calculate recommendations like *"Customers similar to you also bought the following items: ..."*. It is assumed that people will rate the same object differently.

Reputation based Trust Management Systems

Reputation based Trust Management Systems (TMSs) are a soft security mechanism helping to decide whom to trust. Furthermore, they support good behavior, because participants want to get positive feedback, and distract people with bad feedback like dishonest users[33][p. 2]. This way they increase trust in systems like online market places.

A reputation based TMS makes a prediction with a certain probability based on actions in the past to help users making a trust decision. This is how the Web PKI can be enhanced: Users need to get help on decision making based on past experiences. However, they should not automatically trust CA *A* if they trust CA *B* without ever having seen it, as a recommender system would do.

Normally, it is assumed that ratings within a reputation system are independent of a subject's taste[19][p. 221]. For example, people would rate a film differently in a recommender system depending on if they liked it, but in a reputation system, they would rate a film file containing a virus bad independent of their taste. Reputation systems are also called Collaborative Sanctioning (CS) systems[22], since they enable the detection and punishment of misbehavior.

There is a high potential to use CF from recommender systems within a reputation system to improve the reliability of reputation scores – by filtering reputation scores. In this case, it is assumed that there might be some cases where the reputation of the same object differs depending on the subject. This scenario is likely to happen in the Web PKI where users with different ethnical background might have different trust in the same CA.

A popular example for a system combining CS and CF is Amazon. CF is based on similar ratings and users buying the same products. CS is realized by users writing and rating reviews and users rating sellers. Both concepts gear into each other. Amazon can do both – showing users what other users think about a product or seller (reputation) and suggesting other products to buy based on which products were already bought by this user (recommendation).

Trust

Reputation systems are about trust, but trust has many different meanings. Jøsang[22] is using two common notions of trust regarding reputation systems: reliability trust and decision trust.

Reliability trust describes the reliability of something or somebody else.

Decision trust describes to what extent someone is willing to depend on someone or something else in a given situation, even though bad consequences are possible.

Adapted to the CA scenario, decision trust determines if the user is willing to accept a given certificate. The situation might differ which is reflected by the security level (l_{max} to l_{min}) of a website, e.g. online banking has a high security level. In case the certificate was issued by a malicious CA, the established connection might not be secure which is the risk of an user when making a bad trust decision. The CertainTrust[35] model used within the trust view approach is calculating decision trust.

3.2 Basic Reputation System Requirements

According to [33][p. 3], reputation system require long-lived entities to make future predictions from the past, a mechanism to collect and disseminate feedback of current interactions and a way using this feedback to guide trust decisions.

In [33][p. 3], the multiple challenges for reputation system components introduced in section 2.3.1 are mentioned. When **collecting opinions**, there first has to be a sufficient amount of opinions at all, then negative opinions should be elicited such that bad performance is reported and opinions should be honest. **Opinion distribution** has to find a solution for name changes since long-lived entities are required. Furthermore, opinions should be portable, e.g. be able to be distributed between different reputation systems like eBay and Amazon. In a CA reputation system, local trust views are the common format. An user is free to attach multiple services to his trust view. When opinions are **aggregated and displayed**, this has to be done in a way that helps to guide the user's decisions about trust. Since opinions are only considered to speed up bootstrapping and the user is not actively involved in trust decisions where possible, this factor is less important.

3.3 Interoperability Criteria for the Web PKI

A reputation system for CA trust management should:

- Not require modifications of the **existing CA infrastructure** since it is widely adopted.
- Protect from **manipulation** of ratings and the score calculation mechanism, since an attacker could try to manipulate the reputation system instead of attacking the CA directly.
- Meet the **"trust agility"**[27] criteria mentioned in the introduction:
  1) *a trust decision can be easily revised at any time*
  2) *individual users can decide where to anchor their trust*
- Respect **different trust decisions** for the same CA depending on the user's ethnicity to reduce each individuals' attack surface.
- Have a sufficient **availability** to support trust decisions all the time.
- Ensure **privacy**, since browsing histories might contain very sensible information.

- Enable some metric for **trustworthiness** between users despite from the fact that they are strangers.
- Integrate into the existing local **trust view** approach which tremendously reduces the Web PKI attack surface and respects individual trust decisions.
- Have a secure and fast **bootstrapping** mechanism.
- Handle **fake transactions** despite the fact that transactions (connection establishment to a SSL/TLS service) are free of charge and everyone can perform them.

Within the CA scenario, ranking service providers and choosing the best is no option, since typically only one CA signs a certificate for one SSL/TLS service. The input is a certificate and the output guides a trust decision for this certificate.

Some problems *cannot be solved by reputation systems* for CA trust management:

- They don't protect the CA itself from attackers, they only can reveal attacks after they happened. Hardening infrastructure against attacks and training staff against social engineering lies in the CA's responsibility.
- Even though a website is not trustworthy (e.g. if a online shop on this site sells poor quality stuff), the certificate used for connections to this website can have a valid CA signature. In this case, the CA issued the certificate correctly to the website owner and connections to this website are trustworthy, despite the fact the website itself is not. SSL/TLS does not aim at website quality.

3.4 Categorization

This section categorizes existing reputation system approaches and figures out which of them could meet the Web PKI compatibility criteria and gives a basic understanding if they are vulnerable in terms of the attacker model from chapter 2. The goal is to figure out which types of reputation systems could be used to improve the existing Web PKI. Based on these findings, new CA reputation systems are designed in the following chapters 4 and 5. A detailed attack analysis and comparison follows in chapter 6.

Reputation systems can be either built centralized or distributed, both approaches are discussed in the architecture section 3.4.1.

Marti[28] categorizes existing P2P reputation systems, Hoffman[15] analyzes reputation systems in general. They do so by splitting reputation systems into the their components. This thesis is going to use the components introduced in section 2.3.1: collection, calculation and distribution. How to build these components is discussed in section 3.4.2.

3.4.1 Architecture

Centralized

In a centralized architecture, a reputation center collects opinions derived from past transactions and calculates a reputation score for potential future actions[19]. Based on these scores, users can decide whether to transact or not. A centralized reputation systems requires a mechanism to upload (and optionally retrieve) opinions as well an engine deriving issuer trust recommendations.

Privacy and trust

The reputation center is a party trusted by each local user. It knows all users, which opinions they have about CAs and when they were uploaded or updated. It could reveal this information, which is a serious privacy issue.

Users send their opinions to the reputation center, but other users only get the reputation score result calculated by the reputation center. Except from the reputation center, no one has access to their opinions, so it also protects privacy.

Users transitively trust opinions and experiences from other users. The reputation center decides who trustworthy users are, i.e. by a similarity measure. Furthermore, the reputation center could be malicious and return fake reputation scores to users under attack. Users have to trust that the reputation center is calculating issuer trust recommendations correctly.

Implementation

The paper [4] implements a centralized reputation system for trust views, which is described in chapter 4 in more detail.

Distributed

Without reputation center, collecting opinions relevant for an user's decision gets more complicated. There are multiple approaches realizing a distributed reputation system.

Disambiguation

Between the distributed and centralized approach, there is the decentralized approach. Decentralized means that there is not only one central component, e.g. the reputation system is split hierarchically over multiple nodes. In a distributed reputation system, there is no such hierarchy or central component.

Distribution approach

There are several ways to distribute opinion data and evaluate trustworthiness of other users. For example, one could distribute the opinion database over multiple nodes. Using such a distributed database would increase availability and partition tolerance (in case of leaving out the consistency criteria that all nodes see the same data at each point in time, see CAP-Theorem). Therefore, distributing data over a few nodes has some advantages compared to a centralized system. The centralized approach from [4] already distributes opinions over multiple service providers. Even though, this does not change the way how trust is established, since the database runs on multiple nodes but is operated by a single authority.

The goal of a distributed reputation system is to distribute trust from a single authority to many parties. The following approaches meet this criteria.

P2P Distributed reputation systems can be built based on peer-to-peer systems. Other peers are strangers, but their opinions are required to derive a issuer trust recommendation on which a trust decision is based. The challenge is to figure out whose ratings can be trusted without any centralized control mechanism.

Web of Trust Establishing trust based on directly known persons like friends from real life or in a social network or business relationships. This requires a time-consuming bootstrapping process, because an user requires an adequate number of trusted peers until issuer trust recommendations for relevant CAs can be derived.

Structured vs. Unstructured

The overlay network on a P2P network can be formed structured or unstructured[28][p. 2]. In an unstructured P2P network, new users connect to arbitrary peers. Structured P2P networks assign IDs to users, links between peers depend on their ID. Such systems are called Distributed Hash Tables (DHTs).

Performance strongly depends on how the P2P structure looks like. For a reputation system, a fast look-up of as many ratings as possible is required. Furthermore, the P2P network has to handle a lot of churn, since many users are only online for short browsing sessions. Some previously trusted users might rejoin the P2P network or stay for a longer time and it would help if their opinions could be found again.

Common problems and solutions

Opinion distribution and local issuer trust calculation leads to multiple issues. Peers locally compute issuer trust recommendations – no central instance abstracts multiple opinions to a recommendation. Peers need to be able request opinions from other peers, e.g. their neighbors, otherwise they have no information for computation. First, this is allows privacy leakage to other peers. Second, if an attacker knows opinions from neighbors, he can adapt his own opinions. This helps him to appear similar to its well-behaving neighbor, which is a good precondition for launching attacks.

Using pseudonyms to decouple real entities from reputation system users would improve privacy. In a central system, there is only a privacy issue regarding the central component (one Trusted Third Party (TTP)), but within a P2P system, this is an issue among all users. Friedman[10] proposes once-in-a-lifetime identifiers, restricting users changing their identifiers without revealing their true identity. A TTP signs a once-in-a-lifetime identifiers for one user per area. Due to using blind signatures, the TTP doesn't know for which area the identifier was signed. This enables users having multiple pseudonyms which cannot be linked. In [29], another identity management system with similar goals regarding unlinkability within service contexts and prevention of Sybil attacks is proposed. Anyway, pseudonyms don't prevent an attacker to copy ratings in case he doesn't care about the identity of the person. Pseudonyms for a distributed CA reputation system are discussed in more detail in section 5.1.3

A solution to this problem is presented by Ries[34], an approach for privacy preserving computation of trust. Chapter 5 adapts this approach to a distributed reputation system for trust views.

Centralized vs. Distributed

This section gives a short overview which features the centralized and distributed approach have. Solutions for both of them will be proposed in chapter 4 (centralized) and 5 (distributed). A more detailed discussion of differences can be found in the evaluation chapter 6.

Advantages of distribution

- No axiomatically TTP is required.
  In the privacy-preserving computation of Ries[34] the TTP Z is required, but multiple independent instances of Z are possible and Z does not get any information about single opinions.

- Peers can decide whether to share their opinions or not and to whom.
  With the approach from Ries[34] they don't even have to share opinions at all.

- Different weighting of peers depending on their trustworthiness is possible. Compared to the centralized reputation system, users are even free to implement their own selection strategy.

Advantages of centralization

- No data loss if nodes leave the network or are offline for a while.

- Fast look-up of all opinions for issuer trust recommendations.

- Control over all trust views and changes to them, e.g. applying statistical detectors to detect collaborative and unfair opinions as proposed by Yang[38] are easy to implement.

3.4.2 Components

Independent of the underlying architecture, each reputation system has to implement the components introduced in section 2.3.1. All these components have different implementation possibilities.

Collection

First, a reputation system collects user opinions formed by experiences made during the browsing process.

Identities

Each participant needs a unique identity. Identities should be unforgeable, spoof-resistant and in case of system users preserve anonymity[28].

For CAs, these goals are easy to reach, since their certificate signing process is bound to a public/private key pair (spoof-resistance) and launching a new CA identity requires a lot of effort to appear in common root stores (unforgeability). Publishing CA interactions, like which certificate they signed when, is no privacy issue.

For users, anonymity stands in contrast to unforgeability. Reputation systems require long-term identifies to establish trust and be resistant against whitewashing and Sybil attacks. Therefore, a tradeoff between these goals is required, e.g. pseudonyms. However, a TTP is required as soon as users have to establish unique identities, even if this TTP enables users to replace their identities by pseudonyms.

Spoof-resistance can be reached by binding identities to a public/private key pair or a certificate signed by a CA. The first solution is vulnerable in case a MITM is present when the public key is exchanged for the first time, but it does not require a centralized TTP.

Information Representation

The trust model from the existing trust view approach has to be adopted. In this model, information can be single positive or negative experiences or more complex opinions. Anyhow, there exist many other models to represent trust, depending on application requirements.

Information Sources

Information can be *local* like experiences in the trust view approach. Moreover, one can have *trust* relationships to other entities and use their opinions and experiences for issuer trust calculation. External *validation services* can have additional information, e.g. about the certificate history of a service or certificate revocations. Typically, an user checks such external validation services in advance. However, a reputation system could also use their information as source.

Aggregation and Calculation

Based on user feedback, the reputation system calculates reputation scores.

Information Integrity

Depending on the information's source, it might have a different trustworthiness. During calculation, this can be represented by giving these sources different weights or not considering a source below a certain trustworthiness.

In the trust view approach, local experiences have the highest priority. Only when there are not sufficient experiences to calculate an opinion, external services are queried. The reputation system might have multiple sources of information, for example not only a trust view but also contact to a notary server, and weight this information differently. In addition, if there is a measure of trustworthiness amongst users of the reputation system, user opinions can be given a different weight.

Information Conflicts

Opinions might be contradicting. In case one opinion is bad and another opinion is good, the resulting opinion should not be that the overall opinion is somewhat okay. When conflicts appear during opinion aggregation, this should be noted. Within trust views, this is solved by opinions containing a certainty and the conflict aware *cFUSION* operator (see section 5.1.3).

Distribution

Reputation scores have to be disseminated to users helping them making trust decisions.

Information Sharing

A reputation system collects and distributes opinions about CAs. This has to be done in a privacy-preserving and reliable way. An attacker should neither be able to spy on certain users nor to stop certain information to be disseminated.

4 Centralized Approach

This chapter gives a short overview on the centralized reputation system for trust views. The centralized approach is going to be published in [4].

4.1 Architecture

The centralized architecture is shown in figure 4.1.

As in the local scenario there is an entity e_1 with trust view View establishing a connection to another entity e_2. Now, e_1 needs to decide whether the connection is trustworthy.

Additionally, there are other internet users $u_1, ..., u_n$ 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. The network of service providers does not need to be complete, for example, a service provider is not required to trust any other service provider. It is assumed 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.

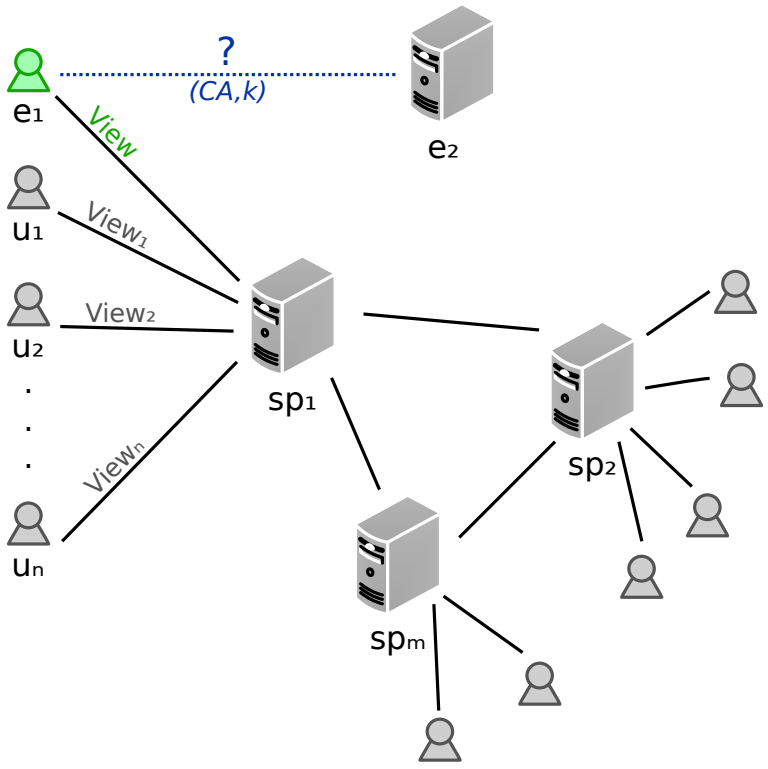

Figure 4.1: Architecture of a centralized reputation system for trust views

4.2 Functionality

The reputation system is providing recommendations for the issuer trust assigned to a CA with key k. Different user opinions about issuer trust are combined with the conflict aware *cFUSION*

operator by sp_1 (see section 5.1.3 for more details on the *cFUSION* operator). In case sp_1 does not have sufficient information, the query is forwarded to another trusted service provider and so forth in order to be able to provide a recommendation.
The process is the following:

1. e_1 establishes a SSL/TLS connection to sp_1 and authenticates itself to sp_1 (e.g. by 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 4.3 for trust view selection strategies in a centralized reputation system).

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*.

The following section describes in detail how to choose trust views for the aggregation step and how to aggregate opinions to a single recommendation.

4.3 Reputation Calculation

First, a basic selection and aggregation strategy is presented. Afterward it is shown how clustering can be applied on this strategy for pre-computation and efficiency improvements.

4.3.1 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[23] and the dependence of the trust views on the subjective browsing behavior as shown in [2], one can follow that trust views reflect an entity's requirements in respect to the relevance of CAs.

Since different entities have different relevant CAs, one can 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 [9].

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 *TA*s contained within trust views and omitting certificates due to privacy issues, the Jaccard similarity index can be applied to trust views as follows. Herein, *TA*s from different trust views are considered 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 *TA*s contained in View_i, $i \in \{1,2\}$ and n is the number of *TA*s shared by the trust views.

Trust view selection process

Let $TV_{(k,ca)}$ be the set of trust views in the service provider's database containing 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 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. This 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. However, a cut off at b has to be applied, otherwise if solely trust views with a low Jaccard similarity are found, this would result in relatively high weights when applying the *cFUSION* operator. The definition of an optimal bound b requires a larger data set than currently available and is due to future work. A legitimate choice could be $b = 0.8$ (according to the evaluation in [4]). Besides that, the similarity weighting and cut off strategy provides protection against Sybil attacks.

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 containing a trust assessment for (k,ca). This can be done more efficiently by clustering trust views in advance and then performing a local search within the cluster that is most similar to the one of the requesting entity.

4.3.2 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. This section explains 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 [12], 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 [18], 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, the K-means parameter is set to $\kappa = \frac{1}{(1-b)}$ 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.

4.3.3 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 4.3.1, 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.

4.4 Improvements

This section proposes improvements to the basic centralized approach. Including them is future work.

Selection strategies

Opinions used to calculate reputation scores should be chosen different depending on whose trust decision they should support. Trust in a CA might differ depending on an user's ethnical background and preferences.

The centralized trust view reputation system from [4] takes the following parameters into account:

- Similarity regarding the number of CAs equal in two trust views (Jaccard coefficient).
- Trust views with less similarity than b are cut off.
- The trust view selection process is accelerated by pre-computed clusters of similar trust views.

Other factors one could take into account are:

- The total number of opinions contained in a trust view.
- Similar voting patterns: not only regard if CAs within the trust view are similar, but also if CAs in the other trust view have similar opinions[15][p. 28].
- Fading memories: keep exponential more information about current transactions[15][p. 23].
- Whitewashing prevention: only take trust views into account that existed for a certain time and were updated regularly.
- History: use history (local and/or remote) to statistically detect opinion changes and take action.
- Usage of correlation and SVD based methods – more accuracy, but computationally expensive.

Limit opinions

In popular reputation systems like eBay or Amazon, each transaction requires money, as well for the buyer (buying an item) as for the seller (transaction fee). An user can only rate another user after a transaction was performed. This tremendously limits the risk of fake opinions. Of course one can cheat this system up to a certain degree, for example an attacker could sell products to known persons without actually giving them the products but having an agreement with them for getting positive ratings. However, the attacker still has to pay a transaction fee making a positive reputation costly.

In the Web PKI, entities can establish secure connections to other entities for free. An user could have a trust view containing many opinions without any cost associated to them. Furthermore, an user has to be able to upload its trust view – otherwise, he could not get any issuer trust recommendations.

To limit the number of opinions an user can upload and that get a high weight during calculation immediately depending on similarity to other trust views, one can introduce a balance. An user starts without having any balance, hindering an attacker from re-entering the reputation system with a new identity to introduce new opinions. The balance gets positive after a certain time interval, if a sufficient amount of opinions matches existing opinions in the reputation system. Only opinions of users with a positive balance are taken into account. In case the opinion would have a high weight, for example because a CA is new and there are only a few opinions about it, the balance has to be higher to consider this opinion during the issuer trust recommendation calculation.

Apply Statistical Detectors

A centralized reputation system has a history of opinions and their changes. One can apply statistical detectors and trust to determine if an attacker is trying to introduce manipulated opinions as described by Yang[38].

This approach is not able to detect *(1)* unfair opinions introduced since the introduction of a CA and *(2)* small but continuing changes in opinions. However, both weaknesses in statistical detectors require an attacker to perform a long-term attack, which is very costly.

5 Distributed Approach

In this chapter, multiple distributed reputation systems are discussed. All proposals have a focus on privacy and attack robustness.

5.1 Encrypted approach

This section describes protocol basics from a privacy-preserving protocol for computation of trust by Ries[34] on which the following CA reputation system is based. After introducing the protocol's properties, protocol primitives are explained on which the encrypted distributed approach is based giving the reader a basic understanding of its security mechanisms. More details and mathematical proofs can be found in the original paper. Problems and solutions for integrating this protocol into a CA reputation system for trust views are discussed.

Despite the centralized approach, in a distributed system, all peers are equal. A peer can have the role of requesting opinions (A) or offering opinions (B) about a service provider (C, in this case a certification authority).

5.1.1 Properties

Adapted to the Web PKI scenario, the approach from Ries[34] has the following properties:

1. An user A is able to calculate the issuer trust recommendation for a CA C.

2. Using Zero Knowledge Proofs (ZKPs), A can check if an other peer B_i is trustworthy. This is possible since A only needs to know if other opinions of B_i were accurate or not.

3. An user A only gets encrypted opinions from other peers B_i, but local calculation can be done by A using homomorphic encryption operations.

4. A TTP Z receives and decrypts issuer trust recommendations calculated by A. Z does not learn about individual opinions, since it only receives the calculation result.

A protocol extension hinders attackers from sending arbitrary issuer trust recommendations from B_i to Z for decryption as well as sending single opinions from B_i to Z for decryption (see section 5.1.4). Since the protocol is based on a Bayesian trust model and users only request the trust value for a single CA, some details in the protocol need to be modified, to integrate the trust view approach.

5.1.2 Architecture

Every node within the network can have the role of a requesting entity A (former: e_1), an user B_i with a local trust view helping A making a decision or be the TTP Z performing the homomorphic decryption part of the protocol.

Assumptions

- Each node has either the role of A, B_i or Z but not multiple roles during one run of the protocol.
- Z doesn't collude with A or B_i.
- Z has to be trusted by A and B_i.
- A and B_i are free to choose a different Z for each protocol run, but during each run, all of them have to agree on the same Z.
- An external identity manager is required to issue Sybil-free identities (for more details see section 5.1.3). Each peer is able to check if the identity of another peer was issued correctly.

Figure 5.1 depicts possible roles of equal peers during one protocol run, the end entity e_2 to which a connection is going to be established and the identity manager.

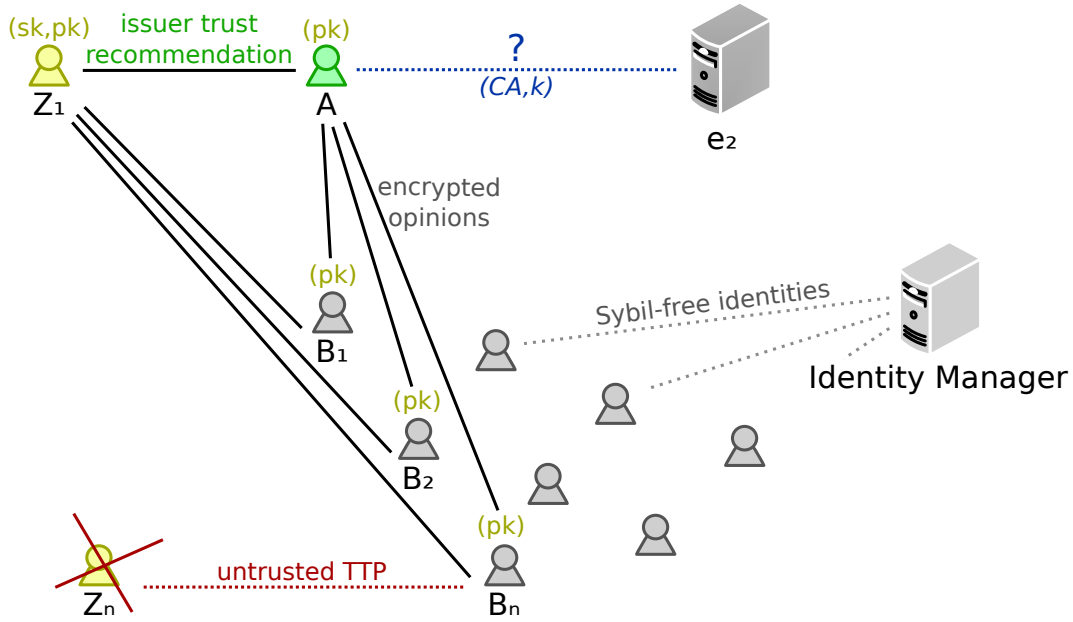

Figure 5.1: Distributed reputation system, encrypted approach: peer roles

5.1.3 Protocol primitives

The following sections introduce basic protocol primitives increasing security and privacy.

Zero Knowledge Proofs

A ZKP proofs a statement without revealing anything else about the statement than if it is true or not[31]. A *proof* in this context is not a mathematical proof: A *prover* wants to convenience a *verifier* about the veracity of a statement.

First, an interactive proof system for a set S is required. It has to be *complete* which means that for every true statement $x \in S$, after an interaction of the verifier with the prover, the verifier always accepts on common input x. In case of a false statement $x \notin S$, the interactive proof

system is *sound*, which is realized by the verifier rejecting with probability $1/p(|x|)$ for some polynomial p and every potential strategy P^*.

Furthermore, these interactive proofs have to be zero-knowledge, which means that a MITM between prover and verifier does learn any information on the set S from watching their communication.

Simple ZKP example: cave

In figure 5.2 a simple zero knowledge scenario is shown. Alice (grey) knows how to open the door in a cave, e.g. she knows a pass code or what the key looks like. Alice wants to proof that she can open the door to Bob (green) but does not want him to see how she does so. Therefore, Alice walks into the cave in a random direction, either towards the A side or towards B side of the door, while Bob is waiting outside. Bob does not know if Alice is on the A or on the B side at this moment. Then he enters the cave and goes up to the fork. Bob tells Alice on which side to return. If Alice returns on the correct path, everything is ok.

In case Alice knows the secret, she is always able to return on the correct path, independent of if she chose side A or B. If she already was on the correct side, she just has to return, otherwise, she opens the door.

In case Alice doesn't know the secret, she has can only return with a probability of $\frac{1}{2}$ on the path requested by Bob. Bob can repeat the test multiple times until the probability gets so small that he is convinced of Alice knowing the secret.

This scheme protects Alice from being eavesdropped while opening the door and using the secret.

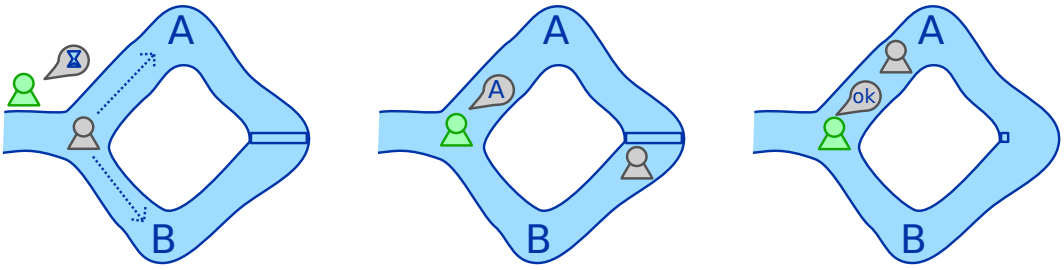

Figure 5.2: ZKP cave

ZKP for proof of opinion accuracy

An user could simply use all available opinions for an issuer trust recommendation calculation. However, this would lead to similar recommendations for everyone independent of personal preferences. Furthermore, an attacker could spread arbitrary trust views containing manipulated opinions with ease to everyone.

Therefore, the centralized approach from [4] introduced a selection strategy. Such a strategy can consider aspects like similarity of trust views to weight opinions. Comparing trust view similarity was one approach to determine if the issuer trust recommendation calculation for an entity e_1 should contain opinions from another user's trust view View_i or not. In case e_1 and u_i have similar trust views, their opinions are likely to have the same tendency. Within the centralized reputation system it was easy to compare each user's trust view without transmitting it to other users, since all information stayed on one central server during computations.

In a distributed system e_1 cannot simply ask multiple users u_i for their complete trust view due to privacy issues. An attacker could just crawl the whole overlay network, request trust view data and launch advanced attacks based on this.

In Ries[34], only encrypted opinions are transmitted and later on homomorphic encryption is used to perform operations on encrypted opinions. To be robust against malicious users altering opinion data while being transmitted, one has to ensure that the encrypted values R and S at e_1 correspond to the cleartext values r and s of the opinion of u_i. Furthermore, the tendency of the opinion can be bound to the encrypted opinion.

1. e_1 has to ensure that the experience information is linked to the encrypted opinion using a ZKP.

2. To evaluate if an opinion was accurate or not, it is only checked if there were more positive (r) or more negative (s) experiences. ZKPs can show if the tendency of an opinion is $r < s$, $r > s$, $r = s = 0$ (no experiences) or $r = s \neq 0$ (for derivation and proof see [34]).

An user A entering the distributed reputation system for the first time has to do many protocol runs with previously known issuer trust recommendation results. Based on these results, A can check if the opinion tendencies of recommenders B_i were accurate to decide who is trustworthy. Instead of calculating the Jaccard value as in the centralized approach (see section 4.3.1), A collects positive and negative experiences about other recommenders B_i. Hence, A does not only have an opinion about CAs it has already seen but also about other peers. This information about recommenders B_i can later on speed up the collection process, if an user only queries trustworthy peers for their opinions.

Trustworthiness: Negative experiences are more severe than positive experiences in the CA scenario. It is normal for a CA to have more positive than negative experiences, even if it is malicious in some cases. One could multiply negative experiences with a constant factor, such that their severeness is represented correctly.

Privacy Attack: Even though the ZKP prevents from transmitting opinions, an attacker could get an idea of which CAs an user has already interacted with. Therefore, if an user gets too many of these requests, it should leave them unanswered or introduce random replies.

Sybil-free Pseudonyms

To establish trust within a reputation, users need unique, long-term identifiers. Such identifiers prevent common attacks like the Sybil attack and other attacks based on it like whitewashing. Despite the first expectation, they do not necessarily stand in contrast to privacy issues like pseudonyms and untraceability.

Sybil-free pseudonyms typically require a TTP issuing initial identifiers. This way, the TTP has the possibility to hinder Sybil attacks when issuing identities, e.g. by using statistical detectors and hardening the registration process with a Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA).

Martucci[29] proposes an identity management system supporting role-based pseudonyms per service context for reputation systems. The idea is that a TTP signs an users unique identity. Afterward, the user is able to issue pseudonyms on its own and proving in zero knowledge these pseudonyms belonging to an identifier were originally issued by the TTP. The proof is made in a way allowing only one pseudonym per service context C^i, the creation of multiple pseudonyms per context can be detected. Pseudonyms cannot be linked over different service contexts by service providers.

Problems within one reputation system

In the CA reputation system, there is *only one service context*. Of course, one could split the reputation system by a criterion like TLDs. Splitting the reputation system into such parts would hinder many possibilities in implementing an appropriate selection strategy, e.g. when partitioning by TLD, one could no longer compare trust view similarity considering all experiences. Each CA signs multiple TLDs. Splitting trust views by CAs would also prevent trust view comparison as part of an appropriate selection strategy. Moreover, one could interpret each connection between two peers as isolated service context with different pseudonyms, causing a tremendous management overhead.

Considering that underlying layers like the P2P and the IP network use constant identifiers per peer, splitting service context on the reputation system overlay shared by all users does not make sense with regard to enhance privacy.

It would be nice to give the user a possibility to *(1)* change his pseudonym after each online session if wanted and *(2)* only contact a TTP once in a lifetime for issuing the initial identity, but not for every session. However, these demands stand in conflict to each other: if unlinkable pseudonyms could be issued autonomously by the initial identity without restrictions, a Sybil attack starting from a single identity and issuing multiple pseudonyms would be possible. Systems like Trusted Platform Modules (TPMs) or the neuer Personal Ausweis (nPA) use very similar solutions, but there the initial identity is bound to hardware and cannot be cloned over multiple systems.

Therefore, in the CA reputation system scenario, users either have to keep their identifier or have pseudonyms issued per session by a TTP. In the first case, users had no pseudonyms at all. In the second case, users have to trust the TTP not to publish their pseudonym changes. Furthermore, the TTP is only allowed to issue one valid pseudonym for an identity at a time to prevent Sybil attacks.

Sybil-free identifiers within one reputation system

A TTP issues unique identifiers. For simplicity, it is assumed that identifiers remain constant during the remaining part of distributed approaches in this thesis.

An identity is free to choose to use a new pseudonym per session issued by the TTP to increase privacy. This causes some overhead since the identity appears new to other peers and therefore the bootstrapping process, e.g. establishing trust with other peers, has to be repeated. Furthermore, the TTP has to ensure that pseudonym validities do not overlap in time, otherwise these pseudonyms could be used for Sybil attacks.

Homomorphic Encryption

Homomorphic encryption means that computations performed on encrypted data persist after decryption. Multiple partially homomorphic encryption schemes exist which are quite efficient. Furthermore, a few fully homomorphic encryption schemes exist but they are way less efficient. Fully homomorphic encryption is still an open field of research with the first approach published in 2009 by IBM[6].

The scheme of Ries[34] is built on Paillier[32] which is based on Rivest, Shamir and Adleman – an asymmetric scheme for cryptography and signatures (RSA) modulus N. Features of Paillier are:

- addition of two variables within $(\mathbb{Z}_N, +)$

- multiplication of a variable with a constant within $(\mathbb{Z}_N^*, \cdot)$

Numbers used in these operations have to be sufficiently small, depending on the size of N.

cFUSION operator

Example

Figure 5.3 depicts the functionality of the conflict aware *cFUSION* operator on a weather example. Combining two contradicting opinions with a high certainty leads to a new opinion with a low certainty.

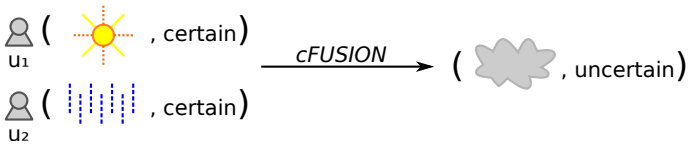

Figure 5.3: *cFUSION* weather example

Operator definition

CertainLogic provides three *FUSION*-Operators for aggregating opinions. The trust view approach is based on the *cFUSION*-Operator [11], 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., [20, 11], do not account for conflict and only average the trust and certainty values of the resulting opinion. In addition, *cFUSION* allows to assign weights to input opinions to give them higher or lower importance. *cFUSION* is defined in [11]:

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:

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

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

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

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

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

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

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

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

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

The CertainLogic *cFUSION*-Operator is commutative.

Continuing the example from above, having two opinions with the highest certainty $c_{A_1} = 1$ and $c_{A_2} = 1$ but the lowest and highest trust value $t_{A_1} = 1$ and $t_{A_2} = 0$, and giving them the same weight $w_{A_1} = \frac{1}{2}$ and $w_{A_2} = \frac{1}{2}$ the result is as follows:

$$t_{\widehat{\oplus}_c(A_1,A_2)} = \frac{1}{2}$$
$$DoC_{(A_1,A_2)} = 1$$
$$c_{\widehat{\oplus}_c(A_1,A_2)} = 0$$

This means, that the resulting trust is in between 1 and 0 at no certainty. The new initial trust f is the weighted average of the old values.

Homomorphic encryption problems

Applying a partially homomorphic encryption scheme like Paillier[32] does not work for the *cFUSION* operator:

- *cFUSION* operates on rational numbers $\mathbb{Q}$ while Paillier operates on integers $\mathbb{Z}$
- *cFUSION* multiplies several variables in a row, but Paillier only supports multiplication of one encrypted variable with an unencrypted constant

To implement a homomorphic *cFUSION* operator, a fast and fully homomorphic encryption scheme would be required.

5.1.4 Privacy-Preserving Computation of Issuer Trust Recommendations

This section describes the protocol for privacy-preserving computation by Ries[34] and shows multiple solutions to integrate it into a CA reputation system.

Basic Protocol

In the following, the basic protocol steps from Ries[34], already a little adapted to the trust view scenario, are explained. Figure 5.4 gives an overview of who interacts with whom.

Adaption

In the basic protocol of Ries[34], a *Bayesian trust model* is used. This trust model needs either to be replaced with the trust view model (an extended Bayesian trust model called CertainTrust[35]) or the trust view model has to be altered to use the Bayesian trust model.

Experience space: Their Bayesian model corresponds to the experience space in trust views, where experiences are divided in positive and negative evidence and have no certainty.

In the protocol from Ries[34], trust computation is performed with a *consensus* and a *discounting* operator on experiences. Both are very simple, such that they can be implemented using the Paillier homomorphic encryption scheme.

Opinion space: The opinion space used in the trust view model to aggregate issuer trust recommendations from multiple users is only existent in the trust view model.

The *cFUSION* of the trust view model operator does both operations from Ries[34]: it builds a *consensus* over multiple opinions and is *discounting* by giving different weighs to opinions, e.g. depending on their trustworthiness. However, using the *cFUSION* operator would require fully homomorphic encryption.

For simplicity, the following protocol steps assume operations on the *experience space*. Further details on how to adapt calculations to the experience space are given in section 5.1.4.

Reputation system integration

Every time when A establishes an SSL/TLS connection to e_2, A needs to know if the connection to e_2 is trustworthy. Only in cases the local trust view does not contain sufficient information to decide whether the connection is $R = trusted$ or $R = untrusted$, external validation services are queried during the *trust validation* process. Then, the consensus of external validation services decides if the connection is $R = (trusted, untrusted, unknown)$.

User interaction is minimized. The average internet user has no possibility to check on his own if a MITM was present during connection establishment. In contrast to the usual buyer and seller scenario, an user is not able to tell if a transaction was good or not just by looking at the price, product and delivery time. Typically, external validation services are contacted automatically to gain information about the certificate and to decide if the associated experience is positive or negative.

In case all validation services fail and R remains $unknown$, the user has to make the trust decision on his own, whether to trust the certificate or not. He is prompted for a decision as browsers do on self-signed certificates.

Querying external services for every connection establishment or asking the user to make a decision on its own takes a lot of time. Furthermore, querying external services is a privacy issue. The goal is to make the local trust view work autonomous, which typically requires collecting ex-

periences over multiple months. For *fast convergence* of the local trust view, asking the reputation system for opinions is required.

The local trust view has the following state before contacting the reputation system:

- The result of an external validation service for the current experience is known ($R = (trusted, untrusted, unknown)$).
- Based on the experience, the user A can already connect to e_2.
- In the background, the reputation system is queried for an opinion about the issuer trust of (CA, k) for fast convergence of the trust view.

0. Setup

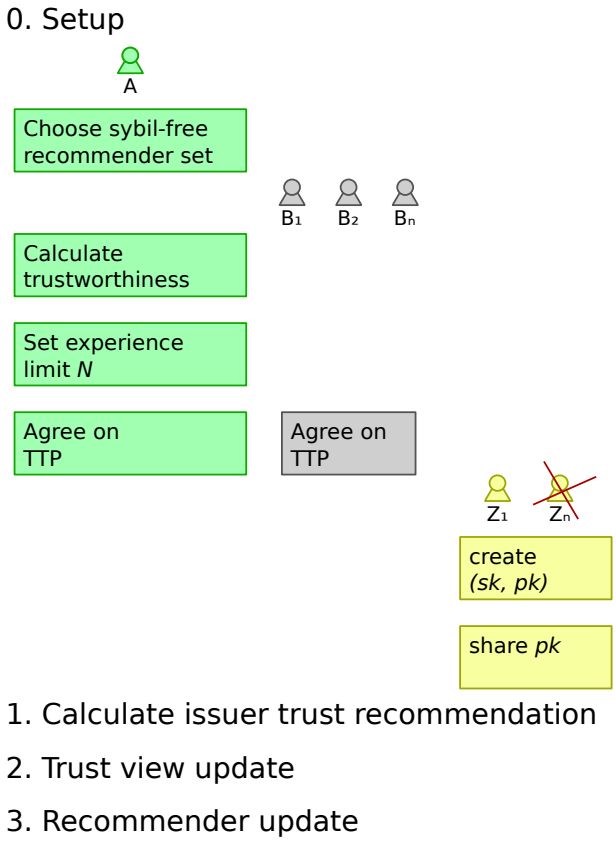

Figure 5.4: Basic Privacy-Preserving Computation of Trust: Setup

Within the setup phase, protocol parameters are initialized and protocol participants are introduced.

- The user A chooses multiple recommenders. The set of recommenders B_i is assumed to be Sybil-free (therefore, A has to check if their identities were issued correctly by the identity manager). For anonymity purposes, the recommender set has to be sufficiently large.
- A calculates the trustworthiness of each B_i (based on previous experiences with B_i).
- A and each B_i have to agree on a TTP Z.
- A informs B_i about the maximum number of experiences N to provide to limit an attacker in providing an arbitrary high number of experiences.
- The TTP Z creates a public key pair (sk, pk) and shares the public key pk with A and B_i.

Selection Strategy: Choosing recommenders B_i and giving them a weight depending on their trustworthiness corresponds to the selection strategy of the centralized approach. Instead of collecting experiences about the accuracy of recommenders, one could also count the amount of equal CAs using the Jaccard similarity index. Recommenders B_i below the lower bound b can be cut off as in the centralized selection strategy in section 4.3.1 to avoid giving untrustworthy recommenders an influence during the calculation process.

1. Calculate issuer trust recommendation

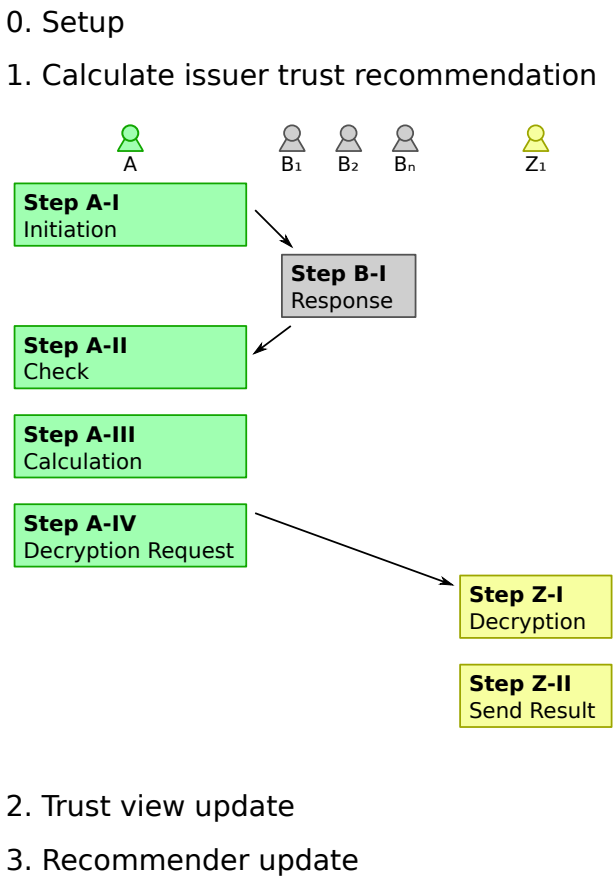

Figure 5.5: Basic Privacy-Preserving Computation of Trust: Calculation

A-I (Initiation) A sends a request for experiences with (CA, k) to each B_i.

B-I (Response) Each B_i answers with experiences encrypted by pk. B_i provides information for ZKPs used to:

- check if the number of experiences provided is below N
- show which one of the tendencies $r < s$, $r > s$ or $r = s$ the experiences have

A-II (Check) A checks the answers from B_i for correctness using ZKP information.

A-III (Calculation) A calculates an *issuer trust recommendation* on the encrypted experiences. In the experience space, the homomorphic Paillier scheme can be used to apply the discounting and consensus operators.
(Alternate calculation methods could be inserted here, e.g. a homomorphic *cFUSION* operator on the opinion space.)

A-IV (Decryption Request) A sends the encrypted calculation result to the TTP Z for decryption.

Z-I (Decryption) Z decrypts the encrypted calculation result using sk.

Z-II (Send Result) Z sends an *issuer trust recommendation* containing positive and negative experiences to A. In case no experiences were made so far, it returns *unknown*.

2. Trust view update

The trust view is updated with a positive or negative experience, depending on R. In case R remained *unknown*, no update is performed.

To speed up the bootstrapping process, once at least one positive experience was made, the *issuer trust recommendation* from the reputation system is adapted into the trust view. Otherwise, the user would have to make many experiences on his own until his trust view would work autonomous.

3. Recommender update

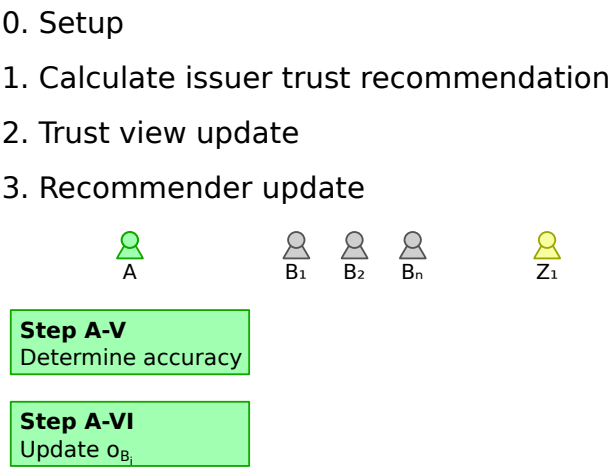

Figure 5.6: Basic Privacy-Preserving Computation of Trust: Recommender update

In the original protocol[34], opinion tendencies ($r < s$, ...) of each B_i are compared to the outcome of a transaction. According to their accuracy, the trustworthiness of recommenders B_i is updated. Opinion tendencies are bound to encrypted opinions which is ensured using ZKPs. For this purpose, there are not only opinions about CAs but also opinions $o^A_{B_i}$, denoting A's opinion about the trustworthiness of a recommender B_i.

Since users can check the outcome of a transaction in advance using external validation services, trustworthiness of recommenders B_i can be evaluated in advance by performing multiple protocol runs and requesting issuer trust recommendations with known results. Furthermore, the trustworthiness of recommenders B_i should be updated after each protocol run, since their trust views could have changed since the last check.

Opinion based calculation

In case there would be a practical way to implement the *cFUSION* operator homorphic, the protocol is very similar to the centralized approach explained in 4.2.

First, a *bootstrapping process* is required.

- A establishes a new identity if it doesn't already have one (or a new pseudonym if wanted).

- A joins the distributed network.
- A checks if the identity of other peers B_i was issued correctly and when.
- A evaluates trustworthiness of other peers B_i based on known issuer trust recommendations doing multiple protocol runs.

The *distributed calculation process* is the following:

1. A sends a pair (k, ca) to sufficiently trustworthy peers B_i to request opinions. The selection strategy S for trust views depends on how trustworthiness is established. Since A doesn't know the complete trust view but only parts of it, selection strategies from the centralized approach don't work. (For using Jaccard similarity, all CAs contained in both trust view must be known).
2. B_i reply with encrypted opinions or *unknown*.
3. A checks the opinions of B_i for correctness.
4. If the number of opinions returned by B_i is $j > 0$ do the following (return *unknown* otherwise):
   a) A aggregates the opinions on the issuer trust with the *cFUSION* operator $\hat{o}_{it}^{ca} = \hat{\oplus}_c(o_{it,1}^{ca}, ..., o_{it,j}^{ca})$ and $\hat{o}_{it}^{ee} = \hat{\oplus}_c(o_{it,1}^{ee}, ..., o_{it,j}^{ee})$. (This requires a homomorphic implementation of the *cFUSION* operator.)
   b) A updates its opinion about the recommender $o_{B_i}^{A}$.
5. A sends the calculation result of $\hat{o}_{it}^{ca}$ and $\hat{o}_{it}^{ee}$ to Z.
6. Z decrypts $\hat{o}_{it}^{ca}$ and $\hat{o}_{it}^{ee}$ and sends the result to A.

Experience based calculation

Assuming that homomorphic *cFUSION* is not possible, one can adapt the CertainTrust model [35] from the trust views to the Bayesian trust model of privacy-preserving computation of trust[34].

Experience space model

Both have an *evidence model* based on a beta probability density function (PDF). Experiences, which are either positive or negative, can be interpreted as binary events. Probabilities of future events can be modeled based on the beta probability distribution $Beta(\alpha, \beta)$. Its PDF is defined as:

$$f(p|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1},$$
$$\text{where } 0 \leq p \leq 1, \alpha > 0, \beta > 0.$$

Opinions consist of positive experiences r and negative experiences s. The notation of an opinion about an entities trustworthiness is $o(r,s)$. An opinions expectation value is its trust mode $t = mode(\alpha,\beta)$.

$$t = mode(\alpha,\beta) = \frac{\alpha-1}{\alpha+\beta-2} = \frac{r+r_0}{r+s+r_0+s_0},$$
$$\text{where } r_0 = s_0 = 1.$$

When establishing a SSL/TLS connection, the feedback $f_{it} \in \{1, 0, -1\}$ of *trust validation* for a certificates issuer trust can be positive ($R = positive$, $f_{it} = 1$), negative ($R = negative$, $f_{it} = -1$) or have no update ($R = unknown$, $f_{it} = 0$). First, feedback is used to update the local trust view. Feedback changes the number experiences for signing end entities $o_{it}^{ee} = (r_{it}^{ee}, s_{it}^{ee})$ and for signing certification authorities $o_{it}^{ca} = (r_{it}^{ca}, s_{it}^{ca})$ within the certification path.

$$o_{it}^{x} = (r_{it}^{x} + 0.5 * (1 + f_{it}^{x}),\ \ s_{it}^{x} + 0.5 * (1 - f_{it}^{x})) \quad \text{if } f_{it}^{x} \in \{1, -1\},$$
$$\text{where } x \in ca, ee.$$

Second, feedback is used to determine the trustworthiness of an user B_i by checking if its opinion was accurate in Ries[34] after each transaction. This is reasonable in a online shop or filesharing scenario, where the user is able to determine if it had a positive or negative experience when products or files arrive at the end of a transaction.

However, in the SSL/TLS scenario, the outcome of an interaction does not directly show if a certificate was issued correctly or not. To find out whether an experience is positive or negative, external validation services are used. Since the result of external validation services typically does not change within a few seconds, they are checked in advance but not afterward.

An user can use feedback of either local trust view computations or external validation services to determine trustworthiness of other users B_i in advance (during bootstrapping). This corresponds to the *selection strategy* already used in the central approach. The feedback $f_i^h \in \{1, 0, -1\}$ rates if the h-th issuer trust recommendation of B_i was accurate. According to this, the opinion o_{B_i} about a recommender can be updated with the same formula as the opinion about the issuer trust.

Both, CertainTrust[35] and privacy-preserving computation of trust[34], use the same *trust propagation* operators. These operators, *discounting* and *consensus*, can be implemented with the homomorphic Paillier encryption scheme.

First, *discounting* is used to process recommendations. In the Paillier scheme, this requires a multiplication of the encrypted recommendation with a locally computed discounting factor.

Having two opinions $o_B^A = (r_B^A, s_B^A)$ and $o_x^B = (r_x^B, s_x^B)$, the opinion of A about x derived via B is defined as:

$$o_x^{AB} = o_B^A \oplus o_x^B = (d_B^A r_x^B, d_B^A, s_x^B),$$
$$\text{where } d_B^A = E(o_B^A) = \frac{r_B^A + r_0}{r_B^A + s_B^A + r_0 + s_0}$$

Second, *consensus* is used for recommendation aggregation. In the Paillier scheme, this requires addition of the number of positive and negative experiences. The consensus operator aggregates two opinions $o_x^{B_1} = (r_x^{B_1}, s_x^{B_1})$ and $o_x^{B_2} = (r_x^{B_2}, s_x^{B_2})$ of B_1 and B_2 about the truth of x. The resulting opinion $o_x^{B_1,B_2} = (r_x^{B_1,B_2}, s_x^{B_1,B_2})$ is the opinion of an imaginary entity who made experiences of both, B_1 and B_2:

$$o_x^{B_1,B_2} = o_x^{B_1} \otimes o_x^{B_2} = (r_x^{B_1} + r_x^{B_2}, s_x^{B_1} + s_x^{B_2})$$

Mapping to the opinion space

The opinion space as used in the *trust view* approach has more complex opinions not only containing the number of positive and negative experiences. There, each opinion consists of a triple (t, c, f).

The value $t \in [0;1]$ represents the trust in the correctness of the statement. As in the experience model, it is defined as the mode of the corresponding beta PDF.

$$t = mode(\alpha, \beta)$$

The certainty $c \in [0;1]$ represents the probability that t is a correct approximation and scales with the amount of collected experiences. The maximum number of expected evidence assuming there is only one context (CA reputation system) is:

$$e = \alpha_{max} + \beta_{max} + 2,$$

where α_{max} and β_{max} fulfill:
$$mean_{coll} := \frac{\alpha}{\alpha + \beta} = \frac{\alpha_{max}}{\alpha_{max} + \beta_{max}} := mean_{max}$$

Under this condition, the certainty is calculated as:

$$c = \frac{f(mean_{coll}|\alpha, \beta) - 1}{f(mean_{max}|\alpha_{max}, \beta_{max}) - 1}$$

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*. In the trust view approach, if no direct or indirect information is known about a certificate, f within o_{it}^{ca} and o_{it}^{ee} is set to 0.5. For more details, see *initialization of trust assessments* in [4].

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

$$E(o) = t \cdot c + f(1 - c)$$

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

Disadvantages of mapping

Within the opinion space, the conflict aware $cFUSION$ operator can be used on plain text calculations in case homomorphic encryption is not required. However, for privacy-preserving computation of trust, homomorphic encryption is required, but using the Paillier scheme, only computations within the experience space are possible and afterward mapping to the opinion space is required.

When doing calculations on the experience space and mapping into the opinion space later, the certainty gets higher (more collected experiences) even though the unaggregated experiences of multiple users might contradict. Since discounting and consensus are calculated on encrypted experiences, there is no chance to be aware of conflicts. Therefore, when performing a mapping from experiences to opinions afterward, there is no chance to see conflicts.

Extended Protocol

The basic privacy-preserving computation of trust protocol has still some vulnerabilities:

1. A can send arbitrary opinions to Z for decryption.

2. A can choose arbitrary weights d within the discounting operator. By setting $d_i = 1$ and $\forall j \neq i.\ d_j = 0$, only the single opinion of B_i would be decrypted by Z without any obfuscation by aggregated values.

3. An attacker can use linear equations on repeated interactions to figure out a single opinions value.

Therefore, Ries[34] proposes multiple protocol extensions. All new and modified steps are listed here.

For issue *(3)*, noise can be introduced by either B_i on single opinions or by Z on the aggregated result.

0. Setup

Within the setup, the following additional steps are required:

- A and all recommenders agree on $r_0 + s_0 = 2$.
- A and all recommenders in B_i agree on the random functions $rand_1(a, b)$ and $rand_2(a, b)$ with $rand_1(a, b) = -rand_1(b, a)$ and $rand_2(a, b) = -rand_2(b, a)$ (in compliance with the Paillier crypto scheme).
- All recommenders agree on a $partner$ function assigning exactly one partner to each recommender.
- A initializes trust in recommenders B_i with $o^A_{B_i} = (0, 0)$.

1. Calculate issuer trust recommendation

B-II (Initiate calculation and send) If B_i never interacted with A, it initializes a counter for A. Using the $rand$ and $partner$ functions, it obfuscates its discounted recommendation information and sends it to Z.

Z-II (Send result) skipped

Z-III (Aggregate and decrypt B_i) Z receives, aggregates and decrypts opinions of B_i.

old Z-I Performed as in basic protocol, required by Z-IV.

Z-IV (Compare and reply) Z compares the aggregated opinions it received by A with opinions from B_i. In case they equal, Z sends the aggregated opinion to A, otherwise it informs A that an error occurred.

2. Update trust view

No changes are required in this step.

3. Update recommenders

A updates trust values into recommenders B_i.

A-V After calculating the accuracy of B_i's opinion, A sends the encrypted accuracy to B_i.

B-III B_i also updates the opinion A has about it ($o^A_{B_i}$).

5.2 Fast approach

The fast approach skips cryptographic methods for performance and complexity reasons but keeps some privacy preserving properties.

5.2.1 Properties

- Protection from crawling opinion data in a large scale.
- A can do efficient computations on opinions of B_i.
- No need for a TTP Z during computations.
- Usage of the conflict aware $cFUSION$ operator possible.

5.2.2 Architecture

The architecture is very similar but does not contain Z that was required for homomorphic cryptography any more. However, a TTP is still needed for issuing Sybil-free identities.

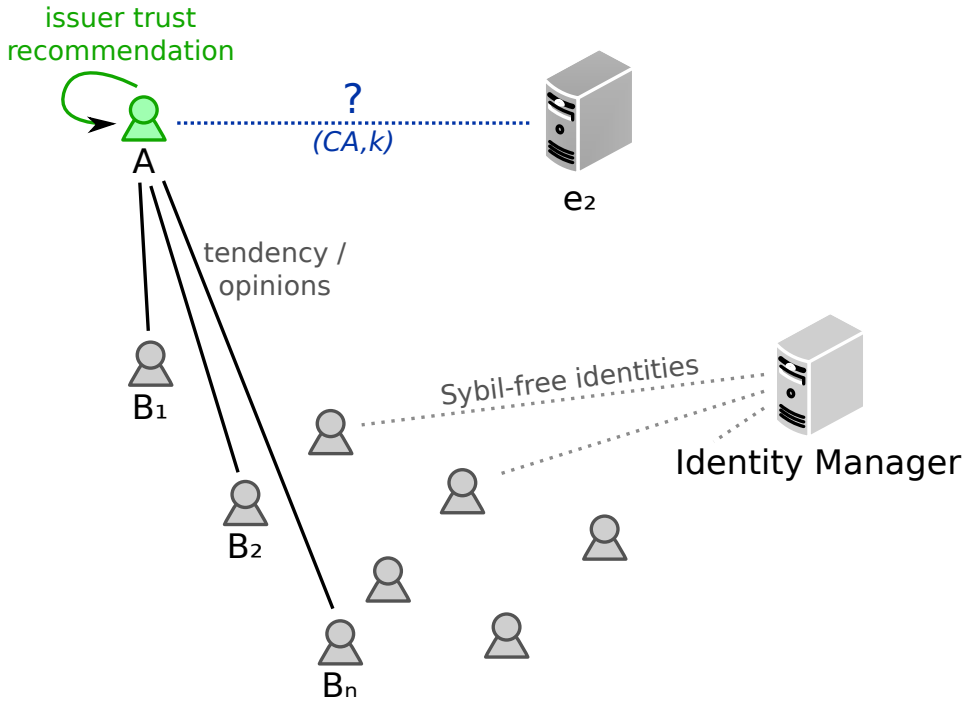

Figure 5.7: Distributed reputation system, fast approach: peer roles

5.2.3 Functionality

As in the fully homomorphic approach based on *cFUSION* introduced in section 5.1.4, the protocol is very similar to the centralized approach explained in 4.2.

First, a *bootstrapping process* is required.

- A establishes a new identity if it doesn't already have one (or a new pseudonym if wanted)

- A joins the distributed network.
- A checks if the identity of other peers B_i was issued correctly and when.
- A evaluates trustworthiness of other peers B_i checking their tendency of selected opinions (if $r < s$, $r > s$, $r = s \neq 0$, $r = s = 0$ but without complete protocol runs and ZKPs). This hinders A from learning too much information about opinions of B_i while establishing trustworthiness.

The *distributed calculation process* is the following:

1. A sends a pair (k, ca) to sufficiently trustworthy peers B_i to request opinions.
2. Recommenders B_i reply with plain text opinions bound to their identity with a signature. In case they get too many requests, they are free not to answer (*unknown*) to protect their privacy. An opinion crawling prevention can be implemented here using statistical detectors.
3. A checks the opinion signatures for correctness.
4. If the number of opinions returned by B_i is $j > 0$ do the following (return *unknown* otherwise):
   a) A aggregates the opinions on the issuer trust with the *cFUSION* operator $\hat{o}_{it}^{ca} = \hat{\oplus}_c(o_{it,1}^{ca}, ..., o_{it,j}^{ca})$ and $\hat{o}_{it}^{ee} = \hat{\oplus}_c(o_{it,1}^{ee}, ..., o_{it,j}^{ee})$.
   b) A updates its opinion about the recommender $o_{B_i}^{A}$.

5.3 Underlying P2P structure

Requirements

In both decentralized solutions, the encrypted and fast approach, a similar underlying P2P network is required.

This network requires structure to find other users and information. Since every user might have information about every CA and does not want to share opinions to every one due to privacy reasons, one can not structure this network by CAs. Every user has its own local repository, its personal trust view.

Each user keeps a list of trustworthy users B_i it contacts when requiring an issuer trust recommendation. The P2P overlay needs to enable users retrieving opinions from known users B_i. Trust views slowly change over time leading to changes considering trustworthiness. Hence users need to refresh information about their contacts regularly.

Assuming that peers rejoin the network under the same identity without using pseudonyms, their node ID within the P2P overlay remains constant. Less identity changes facilitate maintaining the list of B_i.

Node IDs represent identities and are managed by a TTP.

Which P2P overlay to use?

P2P overlays can either be structured or unstructured.

Within an *unstructured* P2P network, connections between peers are established by random. To request an opinion from a known peer B_i, an user has to flood the whole network for finding a route to B_i. This causes a lot of overhead and is not efficient.

To increase efficiency and reduce routing overhead, a *structured* P2P network overlay, also called DHT, should be used. In this case, an user knows where the peer B_i is located within the network structure by its node ID – this process is called key-based routing (KBR).

A possible solution for this is Kademlia, a popular and widespread DHT. It does not only offer a solution for KBR but also a distributed file storage, but this part is not required for the Web PKI and can be skipped. Furthermore, Kademlia implements procedures for joining and leaving nodes.

6 Evaluation

This chapter evaluates functionality and attack robustness of CA reputation systems introduced in this thesis.

6.1 Comparison

In this section, general functionality of CA reputation systems is compared.

	Centralized	**Distributed & Encrypted**	**Distributed & Fast**
Trusted Parties	*direct:* service provider *indirect:* other users and service providers	recommenders B_i, TTP Z, identity manager	recommenders B_i, identity manager
Bootstrapping	• register at service provider • upload trust view	• register identity / pseudonym • join overlay network • find trustworthy recommenders	
Selection Strategy	trust view similarity and clustering	opinion tendency (via ZKP)	opinion tendency (direct)
Issuer Trust Computation Result	external, opinions combined with *cFUSION*	external and local, aggregated experiences	local, opinions combined with *cFUSION*
Computing Time	fast, clustering speeds up similarity search	very slow, homomorphic encryption and many communication steps	fast, possibly slowed down by peers with bad connection
Implementation	easy	complicated	easy
Additional Infrastructure	high-availability service providers	identity manager	identity manager
Churn	no problem	repeat search for trustworthy recommenders	

Table 6.1: Comparison of proposed protocols

6.2 Defense Mechanisms

This section evaluates how robust the CA reputation systems introduced in this thesis are against attacks. It does not give an exhaustive list of possible defense mechanisms for reputation systems in general but takes a closer look at what the CA reputation systems do against attack prevention. Most attacks and defense mechanisms for the centralized approach were already discussed in the corresponding paper[4], but without using the attack model introduced in chapter 2 and containing less details.

Typical defense mechanisms aim at increasing:

- attack duration
- required computing power
- planning complexity

However, attacks planned over a long time and immediately changing behavior cannot always be detected.

Reputation systems also have the side effect of *encouraging good behavior of CAs* since they know they get observed and rated.

Table 6.2 gives an overview of attack robustness, where ++ means that the system is very robust against the attack and – – means that an attacker has a high chance being successful. An explanation on what is done against which attack is given afterward.

	Centralized	Distributed & Encrypted	Distributed & Fast
Unfair Recommendations			
Individuals			
Slandering	++	–	+
Self-Promoting	++	–	+
Random Opinions	++	++	++
Collusion			
Slandering	+	– –	– –
Self-Promoting	+	– –	– –
Random Opinions	++	++	++
Recommendation Reputation	++	+	+
Inconsistent Behavior			
Traitors	+	–	–
Discrimination	–	–	–
Social Engineering	++	+	+
Identity Management			
Registration			
Sybil	++	++	++
Whitewashing	++	++	++
Authentication			
Repudiation	++	++	++
MITM	++	++	+
Impersonation	++	++	++
Resource Availability			
Denial of Service	– – (trust view changes)	– – (slow down each calculation step)	–
Free-riding	++	++	++

Table 6.2: Attack Evaluation

6.2.1 Unfair Recommendations

Detection of unfair recommendations is easier to implement with statistical detectors in a *centralized* system. Yang[38] proposes methods that can detect collaborative unfair ratings and removes dishonest raters. They work under the assumption that an attack happens within a short time interval (a few weeks to a few months).

Individuals

Slandering and Self-Promoting

An attacker can lower or raise the opinions about multiple CAs to stop the reputation system's proper functioning. In the *centralized* a approach this would be prevented by the selection strategy, since the attacker's trust view would no longer be similar to others. Since users don't upload their complete trust view in the *distributed* approach before it is processed due to privacy reasons, malicious recommenders B_i could first answer with an accurate tendency about common

opinions and then introduce false opinions after bootstrapping. This would lead to a positive opinion about the recommender $o^A_{B_i}$ and give false opinions during issuer trust calculation a high weight. In the original approach for privacy-preserving computation of trust[34] introducing false opinions later is prevented by comparing the outcome of an interaction with the opinion of B_i. This is not always possible in the CA scenario, because establishing an SSL/TLS connection does not reveal if the certificate issuer is good or bad. In case a malicious certificate is new and not known by external validation services, the trust decision is left to the user who might not be able to decide properly leading to a false opinion about the recommender B_i.

An attacker can lower or raise the opinion about a specific CA. In all reputation systems, the attacker's trust view would be still very similar to others and therefore the false opinion would be included in the issuer trust recommendation calculation. However, if it is just a single opinion, it does not have a high weight compared to all the other opinions. In case there are only a few opinions, the conflict aware *cFUSION* operator will show there were contradicting opinions for the *centralized* and *distributed & fast* approach. In the *distributed & encrypted* approach, the upper bound N for the number of positive and negative experiences prevents from giving them too much weight, but conflicts are not directly included in the result.

Random Opinions

An attacker can introduce random opinions into its trust view, e.g. to make it appear larger. In all reputation systems, the selection strategy will lower the weight of such trust views.

Collusion

In all reputation systems, due to effective *Sybil* attack defense, the number of identities an attacker can control is limited. This lowers the risk of an successful collusion attack.

Slandering and Self-Promoting

Mechanisms giving each opinion a (limited) weight might not work any more if the amount of false opinions gets too high.

In the *centralized* reputation system, an attacker could upload many trust views with slight variations containing false opinions. This way, the attacker controls more trust views that are possibly similar to trust views of honest users. Since all trust views with a lower similarity than b are cut off, this attack has to be at a very large scale to affect a large amount of users. Increasing b hardens the centralized reputation system. Furthermore, a high trust view upload rate with similar false opinions about certain CAs has a high probability of being statistically detected.

As in the individual false opinion attack, in a *distributed* reputation system, malicious recommenders B_i could manipulate the tendency during evaluation of their trustworthiness. A honest user does not know the overall changes to trust views within the reputation system and hence cannot detect suspicious collusive changes. Even the conflict aware *cFUSION* operator does not help any more if the fraction of malicious recommenders is too large.

Random Opinions

Introducing more random opinions does only slightly change the situation compared to individual random opinions: a good selection strategy still does not take them into account.

Recommendation Reputation

In all reputation systems, trustworthiness is evaluated by the similarity of existing opinions.

Trustworthiness in derived by trust view similarity in the *centralized* approach.

In the *distributed* approaches users do not exchange trust information about each other with other parties, so an attacker has no possibility to introduce conflicting trust information. However, while a honest user is bootstrapping and evaluating the trustworthiness of recommenders B_i, a malicious peer could try to stop messages on the overlay network via a MITM attack stopping honest users from establishing trust to honest recommenders.

6.2.2 Inconsistent Behavior

Traitors

An attacker could oscillate between giving good and bad opinions. The process of giving opinions in this context is very similar to the slandering and self-promoting attacks. Having many oscillating attackers might hinder the reputation system from giving adequate issuer trust recommendations. The faster a reputation system reacts on oscillating behavior or traitors, the better.

In the *centralized* approach, this might cause re-clustering which is described in more detail in the Denial of Service attack in section 6.2.4 and cause a loss of issuer trust recommendation preciseness until clusters are calculated again. The similarity cutoff at b in the selection strategy which is performed on base of trust views within a cluster will prevent from taking non-similar trust views into account during this time.

In both *distributed* approaches, there is a high *reputation lag* until tendencies get exchanged again.

Discrimination

A CA could issue good certificates for most users but discriminate a small subset of users by issuing malicious certificates. In case trust views of this subset of users are more similar to other users but not similar within this subset, the CA could discriminate them and maintain a positive reputation – in all proposed reputation systems. However, one can assume that typically users using a very specialized service are similar to a certain degree.

Social Engineering

All Social Engineering attacks require some kind of interaction with the user that manipulates its behavior. This means that Social Engineering is only possible when an user is able to decide something on his own one could consider as risky behavior.

In all approaches, the attacks described in the attack framework introduction of Social Engineering (see section 2.3.2 are possible. However, the user can be warned when he has to do decisions on experiences on his own or when he is changing the security level parameter l to make him aware of the risk. Indirectly gaining information about an user's browsing behavior is very time-consuming and no large scale attack.

In the *distributed* reputation system, an user needs to join the P2P overlay network. Therefore, he requires a list of entry nodes. An attacker could publish a list of malicious entry nodes or write an e-mail to a specific user and hope for users changing their settings.

6.2.3 Identity Management

All reputation systems use a central component issuing and controlling identities. Within the *centralized* reputation system, the service provider issues and checks identities. Both *distributed* reputation systems use an identity manager issuing identities. Each peer can check if the identity of another peer was issued by this identity manager.

Registration

Sybil

An attacker could try to make the identity management service issue many identities to him within a short time.

First, automatically doing this can be hindered with mechanisms like CAPTCHAs. Other mechanisms could be introduced, e.g. a confirmation via e-mail or even a confirmation that the person is real, however, they go to the cost of the user's privacy. Second, since the identity management service is a central component in all approaches, it can apply detectors like: Were there multiple requests from the same IP or IP range? Is there a very high request compared to normal? Are CAPTCHAs solved wrong very frequent?

All these mechanisms make the registration of new users costly. To undergo the registration procedure, an attacker would require a large botnet, a lot of time and a mechanism to solve CAPTCHAs.

An alternate way for an attacker to have the control over many trustworthy identities would be to control the systems of many honest users participating in the reputation system.

Whitewashing

One can slow down the impact of registering new users (e.g. in the context of a Sybil attack) giving newly registered identities a lower weight when calculating issuer trust recommendations. Assuming there is a grey period in which a CA is compromised but nobody realized that so far, the attacker having the CA under control cannot promote this CA within short time by registering new identities.

In the *centralized* reputation system, it is easy to observe the first upload of a trust view and its change history. Considering the *distributed* approaches, the identity issued by the identity management service should include a date. But even without this countermeasure, it takes some time until a sufficient amount of peers get to know a malicious user and try to figure out its trustworthiness, assuming that users that were part of the reputation system before already have a sufficient amount of trustworthy recommenders B_i.

Moreover, trustworthiness depends on trust view similarity. Therefore, joining the system under a new identity but with the same trust view does not give the attacker any advantages.

Authentication

Repudiation

The reputation system needs to know which user was responsible for which action.

The *centralized* reputation system keeps track of which user uploaded which trust view and when.

In the *distributed* approaches, it is clear which opinion came from whom. The accusation of others for misbehaving is not possible due to the reputation system concept. If a recommender B_i refuses to answer on a request and answers with *unknown*, the source of this answer is clear.

MITM

The goal of a MITM attack is either to breach privacy or to miscommunicate information.

In the *centralized* reputation system, all connections between an user and the reputation system are secured. An attacker cannot perform a MITM attack in this scenario.

The privacy-preserving computation of trust of the *distributed & encrypted* approach uses ZKPs to ensure the tendency of an opinion and the opinion itself belong together which hardens miscommunication. Furthermore, a malicious user A and the TTP Z would have to collude to gain information about opinions of recommenders B_i and breach privacy.

The *distributed & fast* protocol contains less security mechanisms. An attacker can read opinions during transmissions. However, to gain a honest users complete trust view, he would to observe the user over a long period of time. Opinions cannot be modified by an attacker during transmission due to the usage of signatures.

Impersonation

To impersonate another user, an attacker has to steal an user's password (*centralized*) or an user's private key (*distributed*). Both are never transmitted in clear text, thus an attacker requires access to an user's local system to impersonate him or needs to attack the identity management service directly. The protocols themselves are safe against impersonation.

6.2.4 Resource Availability

One has to consider that even in the worst case, when the reputation system is not available for a certain amount of time, this only slows down the convergence of opinions in an users local trust view. The user is still able to ask external validation services to evaluate if an experience was positive or negative.

Denial of Service

An attacker can try to overload the reputation system by introducing new trust views or changing many trust views. In the *centralized* reputation system, this would not only cause load due to database changes but also because recalculating clusters is required. An attacker knowing the clustering algorithm and having a rough idea of which trust views are into the reputation system can make this step last some time. Both *distributed* reputation systems would not be harmed by this because they only exchange information when needed.

An attacker can introduce a delay when asked for opinions. Since the user uploads complete trust views to the *centralized* reputation system, this would not slow down the overall performance. However, especially in the *distributed & encrypted* approach where the exchange of many information is required, this slows down many protocol steps.

Moreover, an attacker could perform an attack on the *distributed* P2P overlay network which would also affect the reputation system. Attacks on the overlay network are out of the scope of this evaluation.

Free-riding

In the *centralized* approach, there is no possibility for free-riding: either the user participates and uploads its trust view or it cannot use the reputation system for receiving issuer trust recommendations.

In a *distributed* reputation system, users are free to ask other peers for opinions without sharing their own opinions. Even if they are not attackers, they might do so to protect their own privacy. In the *encrypted* solution, exact opinion values are not revealed, but an attacker still gets to know with whom the user was interacting.

This is why users should hesitate to share their opinions as attack prevention in a distributed reputation system – in case they get too many requests within short time, they have to block requests to prevent attackers from crawling the complete reputation system. Therefore, free-riding is not necessarily malicious behavior. If an user wants an issuer trust recommendation and some recommenders B_i don't answer, it is still able to do calculations on the remaining information.

Trust Management

The trust view concept enables user-centric trust decisions, tremendously reducing the amount of trusted parties according to user individual needs. All CA reputation systems in this thesis are based on trust views.

The Web PKI is widely adopted and still growing. This means all the CAs and Sub-CAs trusted by current implementations cannot simply be replaced. A potentially larger Web PKI of the future will raise the need for minimizing the number of trust anchors. Reputation systems for trust management in the Web PKI are a promising solution for these problems.

Privacy Enhancements

Whether to choose a centralized or distributed CA reputation approach is mainly a tradeoff between security and privacy. As the attack evaluation showed, the centralized approach is more robust against most attacks. This attack robustness goes to the cost of privacy, since users have to share their complete trust view with the service provider. Moreover, privacy enhancing technologies lead to more complex protocols.

Distributed approaches enable users to decide whom to trust and which information to share in more detail. Applying partially homomorphic encryption, other users don't even learn individual opinions during issuer trust computation. Fully homomorphic encryption required for conflict aware calculations is still an open field of research.

The centralized CA reputation system could also benefit from homomorphic encryption. Opinions could be stored encrypted on sp_1. Since the service provider still knows which opinion belongs to which trust view and which CA, the Jaccard similarity can still be used in the selection strategy. Another service provider sp_2 could interact as TTP for decryption. However, sp_1 could still keep track of trust view changes including involved CAs but not opinion values, which is a privacy issue. Furthermore, colluding service providers could annul encryption.

Usability and User Acceptance

An easy to use user interface is required, which is achievable since users don't have to change many settings to get individual issuer trust decision support.

Loading time of a website is an important factor for user acceptance. Especially the encrypted distributed CA reputation system would introduce a large delay. In the current trust validation process, the reputation system response time is no big issue, since its response is only used to speed up the bootstrapping process, which can be done in background. Response time as perceived by users depends on external validation services, which are queried in advance to check if an experience is positive or negative if nothing about a CA is known. However, if one would consider using CA reputation systems without relying on external validation services at all, issuer trust recommendations have to be calculated fast.

External Validation Service Integration

The CA reputation system could also take queries for single experiences from users and forward them to external validation services. Privacy would be improved by forwarding, since the external validation service could not link the query to an user any more. Again, this privacy improvement comes at the expense of security, since the reputation system could fake experience replies.

In a centralized reputation system, the forwarding party is the service provider. When receiving information from external validation services, the service provider could also use experiences to keep trust views up to date, for example if a CA in some users trust views is known to be hacked and their certificate was revoked.

The forwarding party in a distributed reputation system are trustworthy recommenders B_i. Since their trust views are potentially similar to the requesting user, they might also profit from updated experiences, but not in such a large scale as in the centralized approach. Assuming recommenders are Sybil-free and do not collude, an user could ask multiple recommenders forwarding the experience query to improve security.

Acronyms

CA Certification Authority. 2, 5, 21

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart. 37, 56

CF Collaborative Filtering. 21

CS Collaborative Sanctioning. 21

DANE DNS-Based Authentication of Named Entities. 10

DHT Distributed Hash Table. 25, 51

DN Distinguished Name. 5

DNS Domain Name System. 10

KBR key-based routing. 51

MITM Man in the Middle. 6, 8–10, 12, 13, 18, 26, 36, 41, 55, 57

nPA neuer Personal Ausweis. 38

PDF probability density function. 45, 47

PKI Public Key Infrastructure. 2, 5

RSA Rivest, Shamir and Adleman – an asymmetric scheme for cryptography and signatures. 38

SSL Secure Sockets Layer. 5

TLD Top Level Domain. 10, 11, 19, 38

TLS Transport Layer Security. 5

TMS Trust Management System. 21

Tofu Trust on first use. 9

Tor The Oninon Router. 9

TPM Trusted Platform Module. 38

TTP Trusted Third Party. 25, 26, 34, 37, 42, 49, 50

ZKP Zero Knowledge Proof. 34, 35, 43, 44, 50, 52, 57

List of Figures

List of Tables

Bibliography

[1] M. Alicherry and A.D. Keromytis. DoubleCheck: Multi-path verification against man-in-the-middle attacks. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 557–563, July 2009.

[2] Johannes Braun and Gregor Rynkowski. The Potential of an Individualized Set of trusted CAs: Defending against CA Failures in the Web PKI (Extended Version). 2013. http://eprint.iacr.org/.

[3] Johannes Braun, Florian Volk, Johannes Buchmann, and Max Mühlhäuser. Trust Views for the Web PKI. In *Public Key Infrastructures, Services and Applications*, pages 134–151. Springer, 2014.

[4] Johannes Braun, Florian Volk, Jiska Classen, Johannes Buchmann, and Max Mühlhäuser. CA Trust Management for the Web PKI. 2014.

[5] Comodo. Comodo fraud incident. http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html, Mar 2011.

[6] Michael Cooney. IBM touts encryption innovation, visited Mar. 2014.

[7] 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.

[8] Maurice de Kunder. The size of the world wide web, visited Jan. 2014.

[9] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009.

[10] Eric J. Friedman* and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics & Management Strategy*, 10(2):173–199, 2001.

[11] Sheikh Mahbub Habib, Sebastian Ries, Sascha Hauke, and Max Mühlhäuser. Fusion of opinions under uncertainty and conflict – application to trust assessment for cloud marketplaces. In *TrustCom 2012*, pages 109–118, 2012.

[12] 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.

[13] heise online. Protokoll eines Verbrechens: DigiNotar-Einbruch weitgehend aufgeklärt. http://heise.de/-1741726, Nov 2012.

[14] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium.*, Aug 2012.

[15] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1–31, December 2009.

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

[17] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg 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.

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

[19] Audun Jøsang. Trust and Reputation Systems. In Alessandro Aldini and Roberto Gorrieri, editors, *Foundations of Security Analysis and Design IV*, volume 4677 of *Lecture Notes in Computer Science*, pages 209–245. Springer Berlin Heidelberg, 2007.

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

[21] Audun Jøsang and Jennifer Golbeck. Challenges for robust trust and reputation systems. In *Proceedings of the 5th International Workshop on Security and Trust Management (SMT 2009), Saint Malo, France*, 2009.

[22] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, March 2007.

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

[24] Olaf M. Kolkman and R. Gieben. DNSSEC operational practices. RFC 4641, September 2006.

[25] Eleni Koutrouli and Aphrodite Tsalgatidou. Taxonomy of attacks and defense mechanisms in P2P reputation systems—Lessons for reputation system designers. *Computer Science Review*, 6(2):47–70, 2012.

[26] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, June 2013.

[27] Moxie Marlinspike. SSL And The Future Of Authenticity. In *BlackHat USA 2011*, 2012.

[28] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472 – 484, 2006. Management in Peer-to-Peer Systems.

[29] Leonardo Martucci, Sebastian Ries, and Max Mühlhäuser. Sybil-Free Pseudonyms, Privacy and Trust: Identity Management in the Internet of Services. *Journal of Information Processing*, 2011.

[30] MIT.edu. Web Growth Summary, visited Jan. 2014.

[31] Austin Mohr. A survey of zero-knowledge proofs with applications to cryptography. *Southern Illinois University, Carbondale*, pages 1–12, 2007.

[32] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin Heidelberg, 1999.

[33] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation Systems. *Commun. ACM*, 43(12):45–48, December 2000.

[34] S. Ries, M. Fischlin, L.A. Martucci, and M. Muhlhauser. Learning whom to trust in a privacy-friendly way. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 214–225, Nov 2011.

[35] Sebastian Ries. Certain trust: a trust model for users and agents. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 1599–1604. ACM, 2007.

[36] Sebastian Ries, Sheikh Mahbub Habib, Max Mühlhäuser, and Vijay Varadharajan. CertainLogic: A Logic for Modeling Trust and Uncertainty (Short Paper). In *TRUST 2011*, pages 254–261. Springer, 2011.

[37] Dan Wendlandt, David G Andersen, and Adrian Perrig. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference*, pages 321–334, 2008.

[38] Yafei Yang, Yan Sun, Steven Kay, and Qing Yang. Securing rating aggregation systems using statistical detectors and trust. *Information Forensics and Security, IEEE Transactions on*, 4(4):883–898, 2009.