Maintaining Security and Trust in Large Scale Public Key Infrastructures

Vom Fachbereich Informatik der

Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades

Doktor-Ingenieur (Dr.-Ing.)

von

Dipl. Wirtsch.-Inform. Johannes Braun

geboren in Herrenberg.

Referenten:	Prof. Dr. Johannes Buchmann Prof. Dr. Max Mühlhäuser
Tag der Einreichung:	16.03.2015
Tag der mündlichen Prüfung:	30.04.2015
Hochschulkennziffer:	D 17

Darmstadt 2015

List of Publications

[B1] Johannes Braun. Ubiquitous support of multi path probing: Preventing man in the middle attacks on Internet communication. *IEEE Conference on Communications and Network Security (CNS 2014) - Poster Session*, pages 510–511, IEEE Computer Society, 2014. Cited on pages 52 and 173.

[B2] Johannes Braun, Florian Volk, Jiska Classen, Johannes Buchmann, and Max Mühlhäuser. CA trust management for the Web PKI. *Journal of Computer Security*, 22: 913–959, IOS Press, 2014. Cited on pages 9, 66, 89, and 104.

[B3] Johannes Braun, Johannes Buchmann, Ciaran Mullan, and Alex Wiesmaier. Long term confidentiality: a survey. *Designs, Codes and Cryptography*, 71(3): 459–478, Springer, 2014. Cited on page 161.

[B4] Johannes Braun and Gregor Rynkowski. The potential of an individualized set of trusted CAs: Defending against CA failures in the Web PKI. *International Conference on Social Computing (SocialCom) - PASSAT 2013*, pages 600–605, IEEE Computer Society, 2013. Extended version: http://eprint.iacr.org/2013/275. Cited on pages 9, 32, and 57.

[B5] Johannes Braun, Florian Volk, Johannes Buchmann, and Max Mühlhäuser. Trust views for the Web PKI. *Public Key Infrastructures, Services and Applications - EuroPKI 2013*, vol. 8341 of LNCS, pages 134–151. Springer, 2013. Cited on pages 9 and 66.

[B6] Johannes Braun, Franziskus Kiefer, and Andreas Hülsing. Revocation & non-repudiation: When the first destroys the latter. *Public Key Infrastructures, Services and Applications - EuroPKI 2013*, vol. 8341 of LNCS, pages 31–46. Springer, 2013. Cited on pages 9 and 152.

[B7] Johannes Braun, Moritz Horsch, and Andreas Hülsing. Effiziente Umsetzung des Kettenmodells unter Verwendung vorwärtssicherer Signaturverfahren.

Tagungsband zum 13. Deutschen IT-Sicherheitskongress 2013, pages 347–359. BSI, SecuMedia Verlag, 2013. Cited on pages 9 and 152.

[B8] Andreas Hülsing and Johannes Braun. Langzeitsichere Signaturen durch den Einsatz hashbasierter Signaturverfahren. *Tagungsband zum 13. Deutschen IT-Sicherheitskongress 2013*, pages 565–576. BSI, SecuMedia Verlag, 2013. Cited on pages 9, 10, and 152.

[B9] Johannes Braun, Alexander Wiesmaier, and Johannes Buchmann. On the security of encrypted secret sharing. *46th Hawaii International Conference on Systems Science (HICSS-46)*, pages 4966–4976. IEEE Computer Society, 2013.

[B10] Johannes Braun, Andreas Hülsing, Alexander Wiesmaier, Martin A. G. Vigil, and Johannes Buchmann. How to avoid the breakdown of public key infrastructures - forward secure signatures for certificate authorities. *Public Key Infrastructures, Services and Applications - EuroPKI 2012*, vol. 7868 of LNCS, pages 53–68. Springer, 2012. Cited on pages 9, 136, and 152.

[B11] Johannes Braun, Moritz Horsch, and Alexander Wiesmaier. iPIN and mTAN for secure eID applications. *8th International Conference on Information Security Practice and Experience (ISPEC 2012)*, vol. 7232 of LNCS, pages 259–276. Springer, 2012. Cited on page 158.

[B12] Johannes Braun and Johannes Buchmann. Perfect confidentiality network - a solution for information theoretically secure key agreement. *5th IFIP International Conference on New Technologies, Mobility & Security (NTMS 2012)*, pages 1–5, IEEE Computer Society, 2012.

[B13] Moritz Horsch, Johannes Braun, Alexander Wiesmaier, Joachim Schaaf, and Claas Baumöller. Verteilte Dienstnutzung mit dem neuen Personalausweis. *D-A-CH Security 2012*. syssec Verlag, 2012.

[B14] Detlef Hühnlein, Dirk Petrautzki, Johannes Schmölz, Tobias Wich, Moritz Horsch, Thomas Wieland, Jan Eichholz, Alexander Wiesmaier, Johannes Braun, Florian Feldmann, Simon Potzernheim, Jörg Schwenk, Christian Kahlo, Andreas Kühne, and Heiko Veit. On the design and implementation of the Open eCard App. *GI SICHERHEIT 2012 Sicherheit - Schutz und Zuverlässigkeit*, vol. P-195 of LNI, pages 95–110. Bonner Köllen Verlag, 2012.

[B15] Johannes Braun, Andreas Hülsing, and Alexander Wiesmaier. Schlanke Infrastrukturen für den digitalen Rechtsverkehr - vorwärtssichere Verfahren für qualifizierte elektronische Signaturen. Technical report. TU Darmstadt / ISPRAT e.V., 2012.

[B16] Alexander Wiesmaier, Moritz Horsch, Johannes Braun, Franziskus Kiefer, Detlef Hühnlein, Falko Strenzke, and Johannes Buchmann. An efficient PACE implementation for mobile devices. *6th ACM Symposium on Information, Computer and Communications Security (ASIA CCS 2011)*, pages 176–185. ACM, 2011.

[B17] Johannes Braun, Moritz Horsch, Alexander Wiesmaier, and Detlef Hühnlein. Mobile Authentisierung und Signatur. In Peter Schartner und Jürgen Taeger, editor, *D-A-CH Security 2011*, pages 32–43. syssec Verlag, 2011.

[B18] Johannes Braun, Alexander Wiesmaier TU Darmstadt; Eric Klieme, Linda Strick, and Wolfgang Wunderlich Fokus Fraunhofer. Der elektronische Safe als vertrauenswürdiger Cloud Service. Technical report. TU Darmstadt / Fokus Fraunhofer / ISPRAT e.V., 2011.

Patents:

[B19] Alexander Wiesmaier, Johannes Braun, and Moritz Horsch. EP 2639997 (B1) - Method and system for secure access of a first computer to a second computer. European Patent Office, September 2014. Granted EP Patent 2 639 997.

[B20] Claas Baumöller, Joachim Schaaf, Moritz Horsch, Alexander Wiesmaier and Johannes Braun. EP 2600270 (A1) - Identification element-based authentication and identification with decentralized service use. European Patent Office, June 2013. Pending Application EP Patent 2 600 270.

Acknowledgments

My first thanks go to Professor Johannes Buchmann for giving me the possibility to write this thesis. I thank him for his support and guidance throughout the past five years and for now giving me the opportunity to take over new and interesting tasks within the Collaborative Research Center CROSSING.

Next, I want to thank my collaborators and colleagues that provided valuable comments and helped me to improve this work during many discussions.

Finally, I want to especially thank my family and friends. They supported and believed in me all the time.

To Angelika Braun, my beloved wife.
Without you by my side, this would not have been possible.

Abstract

In Public Key Infrastructures (PKIs), trusted Certification Authorities (CAs) issue public key certificates which bind public keys to the identities of their owners. This enables the authentication of public keys which is a basic prerequisite for the use of digital signatures and public key encryption. These in turn are enablers for e-business, e-government and many other applications, because they allow for secure electronic communication. With the Internet being the primary communication medium in many areas of economic, social, and political life, the so-called Web PKI plays a central role. The Web PKI denotes the global PKI which enables the authentication of the public keys of web servers within the TLS protocol and thus serves as the basis for secure communications over the Internet.

However, the use of PKIs in practice bears many unsolved problems. Numerous security incidents in recent years have revealed weaknesses of the Web PKI. Because of these weaknesses, the security of Internet communication is increasingly questioned. Central issues are (1) the globally predefined trust in hundreds of CAs by browsers and operating systems. These CAs are subject to a variety of jurisdictions and differing security policies, while it is sufficient to compromise a single CA in order to break the security provided by the Web PKI. And (2) the handling of revocation of certificates. Revocation is required to invalidate certificates, e.g., if they were erroneously issued or the associated private key has been compromised. Only this can prevent their misuse by attackers. Yet, revocation is only effective if it is published in a reliable way. This turned out to be a difficult problem in the context of the Web PKI. Furthermore, the fact that often a great variety of services depends on a single CA is a serious problem. As a result, it is often almost impossible to revoke a CA's certificate. However, this is exactly what is necessary to prevent the malicious issuance of certificates with the CA's key if it turns out that a CA is in fact not trustworthy or the CA's systems have been compromised.

In this thesis, we therefore turn to the question of how to ensure that the CAs an Internet user trusts in are actually trustworthy. Based on an in depth analysis of the Web PKI, we present solutions for the different issues. In this thesis, the feasibility

and practicality of the presented solutions is of central importance. From the problem analysis, which includes the evaluation of past security incidents and previous scientific work on the matter, we derive requirements for a practical solution.

For the solution of problem (1), we introduce user-centric trust management for the Web PKI. This allows to individually reduce the number of CAs a user trusts in to a fraction of the original number. This significantly reduces the risk to rely on a CA, which is actually not trustworthy. The assessment of a CA's trustworthiness is user dependent and evidence-based. In addition, the method allows to monitor the revocation status for the certificates relevant to a user. This solves the first part of problem (2). Our solution can be realized within the existing infrastructure without introducing significant overhead or usability issues. Additionally, we present an extension by online service providers. This enables to share locally collected trust information with other users and thus, to improve the necessary bootstrapping of the system. Moreover, an efficient detection mechanism for untrustworthy CAs is realized.

In regard to the second part of problem (2), we present a CA revocation tolerant PKI construction based on forward secure signature schemes (FSS). Forward security means that even in case of a key compromise, previously generated signatures can still be trusted. This makes it possible to implement revocation mechanisms such that CA certificates can be revoked, without compromising the availability of dependent web services. We describe how the Web PKI can be transitioned to a CA revocation tolerant PKI taking into account the relevant standards.

The techniques developed in this thesis also enable us to address the related problem of "non-repudiation" of digital signatures. Non-repudiation is an important security goal for many e-business and e-government applications. Yet, non-repudiation is not guaranteed by standard PKIs. Current solutions, which are based on time-stamps generated by trusted third parties, are inefficient and costly. In this work, we show how non-repudiation can be made a standard property of PKIs. This makes time-stamps obsolete.

The techniques presented in this thesis are evaluated in terms of practicality and performance. This is based on theoretical results as well as on experimental analyses. Our results show that the proposed methods are superior to previous approaches.

In summary, this thesis presents mechanisms which make the practical use of PKIs more secure and more efficient and demonstrates the practicability of the presented techniques.

Zusammenfassung

Public Key Infrastrukturen (PKIs) ermöglichen, mittels der Ausstellung von digitalen Zertifikaten durch vertrauenswürdige Zertifizierungsautoritäten (CAs), die Authentisierung von öffentlichen Schlüsseln. Das ist eine grundlegende Voraussetzung für den Einsatz digitaler Signaturen und Public Key Verschlüsselung. Diese wiederum sind für eBusiness, eGovernment und andere Anwendungen unabdingbar, da sie eine sichere elektronische Kommmunikation ermöglichen. Mit dem Internet als primärem Kommunikationsmedium in vielen Bereichen des wirtschaftlichen, sozialen und politischen Lebens kommt somit der sogenannten Web PKI eine zentrale Rolle zu. Die Web PKI bezeichnet die globale PKI, welche die Authentisierung der öffentlichen Schlüssel von Web Servern im Rahmen des TLS Protokolls möglich macht und damit als Basis für sichere Kommunikation über das Internet dient.

Der Einsatz von PKIs ist in der Praxis jedoch mit vielen bisher ungelösten Problemen verbunden. Zahlreiche Sicherheitsvorfälle in den vergangen Jahren haben Schwachstellen in der Web PKI sichtbar gemacht und dazu geführt, dass die Sicherheit der Internetkommunikation zunehmend in Frage gestellt wird. Im Zentrum der Kritik stehen dabei (1) das global durch Browser und Betriebssysteme vordefinierte Vertrauen in hunderte von CAs, welche den unterschiedlichsten Rechtssprechungen als auch Sicherheitsregularien unterliegen. Dabei genügt es eine einzige CA zu kompromittieren um die Sicherheit der Internetkommunikation zu unterwandern. Und (2) der Umgang mit Revokation von Zertifikaten. Revokation wird benötigt um Zertifikate ungültig zu erklären, wenn diese beispielsweise fehlerhaft erstellt wurden oder der zugehörige private Schlüssel kompromittiert wurde. Nur so kann der Missbrauch durch Angreifer verhindert werden. Revokation ist jedoch nur wirksam, wenn diese auf verlässliche Weise publik gemacht wird, was im Rahmen der Web PKI ein schwieriges Problem darstellt. Desweiteren ist hier die enorme Abhängigkeit einer Vielzahl von Services von einzelnen CAs ein schwerwiegendes Problem. Dadurch ist es häufig nahezu unmöglich das Zertifikat einer CA zu revozieren und damit zu verhindern, dass mit dem entsprechenden Schlüssel weiterhin gültige Zertifikate ausgegeben werden können. Genau das ist jedoch notwendig wenn sich herausstellt,

dass eine CA tatsächlich nicht vertrauenswürdig ist oder ihre Systeme von einen Angreifer kompromittiert wurden.

In dieser Arbeit wenden wir uns daher der Fragestellung zu, wie sichergestellt werden kann, dass die CAs, denen ein Internetnutzer vertraut auch tatsächlich vertrauenswürdig sind. Ausgehend von einer detailierten Problemanalyse stellen wir Lösungsverfahren für die einzelnen Teilprobleme vor. Dabei ist Umsetzbarkeit und Praktikabilität von zentraler Bedeutung. Aus der Problemanalyse, welche die Untersuchung bisheriger Sicherheitsvorfälle und bereits existierender Lösungsansätze beinhaltet, werden Anforderungen für die Problemlösung abgeleitet.

Zur Lösung von Problem (1) führen wir nutzerzentriertes Vertrauensmanagement für die Web PKI ein. Dies erlaubt es die Anzahl der als vertrauenswürdig betrachteten CAs nutzerindividuell auf einen Bruchteil der ursprünglichen Menge zu reduzieren. Damit wird das Risiko auf eine CA zu vertrauen, die tatsächlich nicht vertrauenswürdig ist, signifikant reduziert. Die Beurteilung der Vertrauenswürdigkeit von CAs erfolgt dabei individuell und evidenzbasiert. Darüber hinaus erlaubt uns das Verfahren, den Widerrufsstatus der nutzerrelevanten Zertifikate zu überwachen und somit ein Teilproblem von (2) zu lösen. Die Lösung ist im Rahmen der bestehenden Infrastruktur ohne Performanz- oder Nutzbarkeitseinbußen umsetzbar. Desweiteren stellen wir eine Erweiterung mittels Online-Diensteanbietern vor. Damit ermöglichen wir die lokal gesammelten Vertrauensinformationen anderen Nutzern zur Verfügung zu stellen und damit die notwendige Initialisierung des Systems zu verbessern. Darüber hinaus wird ein effizienter Erkennungsmechanismus für nicht vertrauenswürdige CAs realisiert.

Bezüglich des zweiten Teilproblems von (2) stellen wir eine PKI Konstruktion vor, welche robust gegenüber der Revokation von CA Zertifikaten ist. Dies wird über den Einsatz vorwärtssicherer Signaturverfahren erreicht. Vorwärtssicherheit bedeutet, dass selbst im Falle einer Kompromittierung des privaten Schlüssels, zuvor erzeugten Signaturen weiterhin vertraut werden kann. Damit lässt sich ein Widerrufsmechanismus umsetzen, der es erlaubt CA Zertifikate bei Bedarf zu revozieren, ohne die Erreichbarkeit der abhängigen Web Services zu gefährden. Wir zeigen, wie die Web PKI unter Berücksichtigung der relevanten Standards in eine solch robuste PKI überführt werden kann.

Die in dieser Arbeit entwickelten Mechanismen erlauben es uns, zusätzlich das verwandte Problem der Nichtabstreitbarkeit digitaler Signaturen anzugehen. Nichtabstreitbarkeit ist ein wichtiges Sicherheitsziel für viele eBusiness und eGovernment Anwendungen und erfordert im allgemeinen den langfristigen Gültigkeitserhalt digitaler Signaturen. Dies wird von PKIs nicht standardmäßig unterstützt und erfordert bisher aufwändige Zusatzmaßnahmen wie Zeitstempel durch vertrauenswürdige Zeit-

stempeldiensteanbieter. Wir zeigen in dieser Arbeit, wie Nichtabstreitbarkeit als Standard-Eigenschaft von PKIs etabliert und somit auf Zeitstempel verzichtet werden kann.

Die in dieser Arbeit vorgestellten Verfahren werden im Hinblick auf Praktikabilität und Performanz evaluiert. Dies basiert zum einen auf theoretischen Resultaten zum anderen auf experimentellen Ergebnissen und Analysen. Unsere Ergebnisse zeigen, dass die vorgestellten Verfahren bisherigen Lösungsansätzen überlegen sind.

Zusammenfassend stellt diese Arbeit Mechanismen zur Verfügung, welche den praktische Einsatz von PKIs sicherer und effizienter gestalten und belegt die Praktikabilität der vorgestellten Verfahren.

Contents

1 Introduction

Electronic communication has become an integral part of daily life. In 2015, a monthly traffic of more than 83 exabyte of data being sent over the Internet will be reached [178]. The Internet offers a nearly inconceivable mass of applications to its users such as e-commerce, e-banking, e-government and online social networks just to name a few. As of mid 2014 more than 3 billion Internet users [135] use the services provided by more than 1 billion hosts [134] all around the world. Looking at the different growth rates concerning Internet usage, it is safe to predict that this is by far not the end.

Because of its importance, Internet communication must be protected. Most of the applications mentioned above typically involve sensitive data, for example credit card numbers, medical data or industrial secrets. Security of communication not only refers to the protection of business assets and monetary values but also to the protection of the privacy of the communication partners. And at least since the disclosures of Edward Snowden [124] it is commonly known, that the attackers which aim at the interception, surveillance and the control of the communication are not limited to criminals and totalitarian regimes but also comprise intelligence agencies and other governmental organizations even of democratic countries. In many cases such attackers have a big budget and far-reaching capabilities to intercept the communication channels of the Internet.

The protection of the communication requires: authentication, integrity and confidentiality. This means, the communication partners must be sure with whom they communicate (authentication), that the sent data has not been changed on the way from the sender to the receiver (integrity) and that no unauthorized third party can access the communication (confidentiality). Secure Internet communication is realized using the Transport Layer Security (TLS) protocol, which relies on digital signatures and in many cases public key encryption. Digital signature schemes as well as public key encryption schemes belong to the field of public key cryptography and require key pairs: a private key, which is only known to the owner of the key pair and a public key, which must be provided to any potential communication partner.

In digital signature schemes, the key owner uses the private key to sign messages, and the public key is used to verify the correctness of the signature. In encryption schemes, the public key is used to encrypt messages and the private key is used to decrypt the ciphertexts.

A necessary condition for security of digital signatures and public key encryption is the possibility of public key authentication, i.e., the assertion to which entity a public key belongs. Without this possibility, an adversary can sign messages in the name of any other entity by pretending that his own public key belongs to that entity. Or, in the case of encryption there would be no guarantee, that the entity which is able to decrypt a ciphertext is actually the entity which is intended to be able to do this. The problem with authentication is that the communication partners must authenticate the public keys of each other before they can use digital signatures and public key encryption for the establishment of secure authenticated and encrypted communication channels. But, when considering global communication, it must be assumed that the communication partners have never met in person or communicated through an a priori secure channel which would allow the authentication of their public keys.

This authentication problem is solved with Public Key Infrastructures (PKIs). PKIs reduce trust in the authenticity of a public key of an entity to the trust in the authenticity of a Certification Authority's (CA's) public key and the trustworthiness of this CA. In fact, in PKIs chains of trust reductions that involve several CAs reduce the trust in the authenticity of a public key to the authenticity of the public key and the trustworthiness of a Root CA which serves as a trust anchor. CAs are trusted third parties that issue public key certificates which bind public keys to their owners. Certificates are electronic documents that contain the public key together with the key owner's name and are digitally signed by the CA. By issuing certificates, the CA guarantees for the authenticity of public keys. Furthermore, CAs are responsible for the revocation of certificates, which ends the binding between a public key and an identity established through a formerly issued certificate. Revocation is for example necessary when the respective private key has been compromised in order to prevent misuse of the key in the name of the legitimate key owner.

The certificates are used to authenticate the public keys of the communication partners and subsequently the communication partners themselves. In TLS mainly unilateral authentication is applied, i.e., only the web servers are authenticated by certificates.

This raises the important question which CAs are in fact trustworthy to issue certificates. In recent years, it has repeatedly happened that supposedly trustworthy CAs issued fraudulent certificates to attackers and showed serious breaches of duty

in regard to the handling of security incidents. This enabled attackers to spoof web servers and intercept private communication of Internet users.

Currently, the decision which CAs are to be considered trustworthy is left to operating system and browser vendors. Operating systems and browsers contain root stores: lists of Root CAs which are directly trusted. CAs themselves can delegate trust to subordinate (Sub) CAs making them trusted by operating systems and browsers as well. This is done by issuing CA certificates to the Sub CAs. For technical details on how this is realized we refer the reader to Sections 2.2 and 3.1.3.

The PKI which comprises all these trusted CAs is referred to as the Web PKI. Currently, the Web PKI consists of approximately 1,590 CAs which are controlled by 683 private as well as governmental organizations and are located in 57 different countries [17]. There are several issues why the Web PKI fails to provide the desired security which we highlight in the following:

- The huge number of CAs together with the fact that the security relies on the weakest link of the system has disastrous consequences for security: Each of the CAs is equally trusted. Thus, an attacker controlling a single one of the CAs may impersonate arbitrary entities. This is especially problematic because:
  - Any CA must be seen as being ultimately fallible, e.g., due to implementation errors, poor operational practices or human errors.
  - There is no globally standardized mechanism that ensures the trustworthiness of CAs.
  - Governments, which were identified as potential attackers, have in many cases control over CAs. Either the CA is being operated by a governmental organization or access may be obtained through jurisdiction of the country in which the CA resides.
- Revocation of certificates in general and in particular the revocation of CA certificates is problematic.
  - The provision of revocation information to billions of users has shown to be a particular challenge in the Web PKI. However, if the availability of revocation information cannot be guaranteed, this may render the whole revocation mechanism useless.
  - CA certificates can often not be (immediately) revoked, although a CA compromise or CA misbehavior would actually make this step necessary.

> In many cases the authentication of thousands of service providers depends on a single CA. Upon revocation of such a CA's certificate all these services would become unavailable. This makes the removal of faulty CAs problematic if not impossible.

Therefore, the research goal which this thesis addresses is:

To ensure that the CAs which an entity trusts in are actually trustworthy.

Our contribution to achieving this goal will take into account the following critical success factors imposed by the global scale of the Web PKI:

- Scalability: The solutions must scale to billions of users and services and, at the same time, must not depend on being implemented globally.
- Usability: In general, the users are non-experts and cannot be expected to understand the functioning of PKIs.
- Backward compatibility and deployability within the existing infrastructure.

In this work we introduce user-centric CA trust management. It addresses the research goal by individually reducing the number of trusted CAs. The reduction is based on evidence of the trustworthiness of CAs. This significantly reduces the risk to rely on a CA, which is actually not trustworthy. Furthermore, user dependent data collection allows to continuously monitor the revocation status for certificates relevant to a user.

Secondly, we present a CA revocation tolerant PKI construction based on forward secure signature schemes (FSS). The forward security property of FSS maintains the validity of signatures generated prior to a key compromise. Making use of this property, revocation mechanisms can be implemented such that the revocation of CA certificates becomes feasible without being restricted by availability requirements.

We show that these solutions solve the above mentioned issues without introducing significant overhead.

The techniques developed in this thesis that allow for forward security in PKIs also enable us to address the security goal "non-repudiation" which is not guaranteed by standard PKIs. It refers to an entity not being able to deny being the origin of a certain piece of data. For example, non-repudiation is important in many e-business and e-government contexts. Non-repudiation requires long term verifiability of digital signatures. We show how non-repudiation can be made a standard property of PKIs.

In the following, the contributions are summarized and the organization of this thesis is presented.

Contribution and outline

The main contributions of this thesis are:

1. Concept and realization of user-centric CA trust management which enhances the trustworthiness of CAs in the Web PKI.

2. Techniques that enable the Web PKI to tolerate CA certificate revocation and provide non-repudiation.

The contributions are divided into several parts as described in the following.

Chapter 2 – Background In this chapter, the relevant background for this thesis is provided. This includes relevant definitions regarding public key cryptography and an introduction to hierarchical PKIs. We also provide background on Internet communication and introduce computational trust. In particular, we present CertainTrust, the trust model which is used throughout this thesis to represent trust.

Chapter 3 – The Web PKI requires user-centric CA trust management In this chapter we provide an in depth analysis of the Web PKI and a problem exposition. The practical relevance of the identified problems is demonstrated based on an analysis of past security incidents regarding the Web PKI. Also, previous scientific work as well as practical proposals for the solutions regarding the described problems are reviewed and weaknesses are identified. This enables us to identify challenges and requirements for a practical solution.

Starting from this, we define the user-centric trust model for the Web PKI meaning that trust decisions are individualized and trust settings regarding CAs are tailored to the user-specific requirements of a relying entity. Based on a user study, we show the potential of user-centric CA trust management in regard to the reduction of the attack surface of the Web PKI.

Chapter 4 – CA-TMS: User-centric CA trust management Based on the findings in Chapter 3, we develop CA-TMS: a CA trust management system that realizes the user-centric trust model. The core of CA-TMS are trust views that serve as a local and user dependent knowledge base for trust decisions. Trust views achieve two goals: The number of CAs a user trusts is reduced. This leads to a reduction of the attack surface by more than 95%. Secondly, for the trusted CAs the revocation status can be continuously monitored, which enables a reliable provision of revocation information.

An important aspect of CA-TMS is, that it realizes user-centric trust management without involving the user into decision making. This is achieved by means of computational trust along with learning processes and automated trust decisions based on defined decision rules. CA-TMS exclusively builds on data which either is initially available or is collected over time. In contrast to existing proposals, CA-TMS does not require recommended trust values embedded into certificates or the evaluation of certificate policies and expert opinions.

Techniques of previous proposals like public key pinning and certificate notaries are integrated and combined as building blocks. These different building blocks complement each other such that it allows to overcome scalability and usability problems of previous proposals.

Chapter 5 – Service providers for CA-TMS CA-TMS as presented in Chapter 4 protects relying entities from fraudulent certificates issued by CAs that are not part of their trust view or do not fulfill the trust requirements for a specific application. CA-TMS works autonomously and does not require an additional check of every (new) certificate once the trust view is bootstrapped.

The extension of CA-TMS by service providers solves two specific issues: Bootstrapping in case no or only limited input data is available and protection in the face of CAs that are already considered to be trustworthy but suddenly change their behavior and become untrustworthy.

The bootstrapping problem is solved by a reputation system. It makes the knowledge of other relying entities available to a relying entity whenever its own experience is insufficient for decision making. With this mechanism, we speed up the bootstrapping process. The second problem is solved with a push service. CA-TMS implements a function which detects suspicious certificates. The service providers monitor the trust views collected within the reputation system's database. Upon detection of a CA becoming untrustworthy, this information is pushed to all clients whose trust view contains the CA in question.

Chapter 6 – CA revocation tolerant PKI In this chapter we show how to transform a PKI into a CA revocation tolerant PKI. This is achieved by the application of forward secure signature schemes. The special properties of FSS allow the adaptation of certificate validation and revocation mechanisms such that CA certificates can be revoked in a secure way without invalidating former signatures. This prevents the unavailability of dependent services.

Besides providing the concepts, we describe how to implement the solution in the Web PKI. The results show that the implementation is feasible without infrastruc-

tural changes and is covered by current standards.

The performance evaluation includes a comparison to the standard setup where common signature schemes like RSA or (EC) DSA are used. It turns out, that a CA revocation tolerant PKI admits good performance in regard to runtimes and certificate as well as signature sizes.

Chapter 7 – Providing non-repudiation and long term verifiability In contrast to authentication scenarios, in use-cases that require non-repudiation, digital signatures must remain valid and verifiable for a long time. A common example is contract signing.

This solution is a significant improvement of the current solutions which are based on time-stamps generated by trusted third parties. In order to guarantee non-repudiation, time-stamps must be applied to each signature directly after signature generation. This is costly and therefore has prevented the broad application of digital signatures as a replacement for handwritten signatures so far.

Extending the mechanisms used in Chapter 6, in Chapter 7, we provide a solution based on FSS which establishes non-repudiation as an inherent guarantee provided by the PKI. This guarantee is preserved as long as the used signature scheme is considered secure. Additionally, we present possibilities to prevent the sudden break down of the security of signatures based on special properties of XMSS [12], a hash-based FSS.

Chapter 8 – Conclusion Finally, in this chapter we conclude our thesis and discuss future research directions.

2 Background

In this chapter we present the necessary background for the work at hand. We provide the definitions and concepts used throughout this thesis. We start with an introduction to public key cryptography and cryptographic building blocks in Section 2.1. Afterwards, we describe the components and processes of public key infrastructures in Section 2.2. This is followed by a short introduction to secure Internet communication in Section 2.3. In Section 2.4, computational trust models are introduced and in particular CertainTrust, which is used to represent trust in this thesis. Finally, further related work is listed in Section 2.5.

This chapter contains revised and extended parts of the background sections published in [B2, B4, B5, B6, B7, B8, B10].

2.1 Public key cryptography

The security goals relevant to this thesis are integrity, confidentiality, authentication, data authenticity and non-repudiation. Integrity refers to the fact, that data has not been modified since its generation. Confidentiality denotes the property, that data is not available to an unauthorized entity. Authentication is the process of confirming the identity of some other entity. Data authenticity refers to the determination of the originator of data. Non-repudiation is the property, that an entity cannot deny to have performed a certain action, such as having signed a document.

These security goals can be and are in practice achieved with public key cryptography. In general, public key cryptography requires key pairs: a private key (also called secret key) sk that is only known to the owner of the key pair as well as a public key pk, which must be provided to the other participants in a cryptographic protocol. Furthermore, it is generally required to ascertain a relation between a public key pk and the identity of its owner, i.e. to verify the authenticity of a public key. This shows a dilemma: in order to enable authentication, authenticated credentials need to be available to the participants. This also shows the central role of authentication in cryptography.

The authentication of public keys is achieved with public key infrastructures which are explained in Section 2.2. First, several cryptographic mechanisms will be explained in more detail.

2.1.1 Hash functions

Hash functions are important cryptographic building blocks. Generally, a hash function $h : \{0,1\}^* \to \{0,1\}^n$ maps bit strings of arbitrary length $\{0,1\}^*$ to bit strings of length $n \in \mathbb{N}$. $h(x)$ can efficiently be computed for any $x \in \{0,1\}^*$.

To be used in cryptographic applications, hash functions require special properties. These are one-wayness, second-preimage resistance and collision resistance. We only give an informal description of the properties and refer the reader to [34] for formal definitions. One-wayness denotes the property that it is infeasible to invert the hash function, i.e. it is infeasible to compute $x \in \{0,1\}^*$ such that $h(x) = s$ for a given $s \in \{0,1\}^n$. Second-preimage resistance means, that it is infeasible to find a collision for a given $x \in \{0,1\}^*$, i.e. to find an $x' \in \{0,1\}^*$ with $x \neq x'$ such that $h(x) = h(x')$. Finally, collision resistance denotes the property that it is infeasible to find any collision, i.e. it is infeasible to find any pair $x \in \{0,1\}^*$ and $x' \in \{0,1\}^*$ with $x \neq x'$ such that $h(x) = h(x')$. Hash functions that are collision resistant are also called cryptographic hash functions. Collision resistance is for example required to generate digital signatures which will be described in Section 2.1.2.

Collision resistance is the strongest of the explained security properties. Collision resistance implies second-preimage resistance which in turn implies one-wayness, however this does not hold the other way around. A one-way function can be used to construct a second-preimage resistant hash function. But, collision resistant hash functions are not known to be constructable from one-way functions nor from second-preimage resistant hash functions [34, B8].

Another security property of hash function families is pseudorandomness. It means, that a function randomly drawn from a pseudorandom function family is, concerning its in- and output behavior, indistinguishable from a function randomly drawn from all functions with the same in- and output sets. Pseudorandom function families can as well be constructed from one-way functions. Thus, again the collision resistance is a stronger security assumption than pseudorandomness [34].

2.1.2 Digital signature schemes

A digital signature scheme $\mathsf{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ is given by three algorithms:

KGen The key generation algorithm, on input of security parameter 1^λ, $\lambda \in \mathbb{N}$

generates a key pair $(\mathsf{sk}, \mathsf{pk})$ consisting of a secret signing key sk and a public verification key pk.

Sign The signing algorithm, on input of a secret key sk and a message $m \in \{0,1\}^*$, outputs a signature σ.

Ver The verification algorithm, on input of public key pk, a message m, and a signature σ, outputs 1 iff σ is a valid signature on m under pk, else 0.

S is correct if for all $\lambda \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $m \in \{0,1\}^*$, and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, $\mathsf{Ver}(\mathsf{pk}, m, \sigma)$ returns 1 as output.

In general, a message $m \in \{0,1\}^*$ is not signed directly, but the according hash value $h(m)$ is used as input for Sign. Where h is a collision resistant hash function.

When referring to a signature scheme, we refer to a secure digital signature scheme. Secure means, that an adversary is unable to forge signatures without knowing the private key. For formal definitions of the security of signature schemes like existential unforgeability under chosen message attacks (EU-CMA), we refer the reader to [25, 34].

In the following, we introduce key evolving signature schemes and forward secure signature schemes and explain their specific properties.

Key evolving signature schemes

In contrast to conventional signature schemes, in key evolving signature schemes (KES) the secret key sk changes over time, while the public key pk remains the same. The lifetime of a key pair is split into several intervals, say t. These intervals can be defined in different ways. Either an interval corresponds to a time period, e.g., one day. Or an interval ends after the key was used to create a certain number of signatures. Note that this implies that the length of two intervals might differ. Especially, it is possible to associate the intervals with single signatures.

The number of intervals t becomes a public parameter of a KES and is taken as an additional input by the key generation algorithm. A KES key pair has t secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_t$; one secret key for each of the t intervals. The key generation algorithm outputs $(\mathsf{sk}_1, \mathsf{pk})$, where sk_1 is the first secret key. To obtain the subsequent secret key, a KES has an additional key update algorithm KUpd, which updates the secret key at the end of each interval. The signing algorithm takes as additional input the index of the current interval. This index also becomes part of the signature and is therefore available for the verification algorithm. Finally, if a user generates a valid key pair and the key update algorithm is called at the end of each interval, then a

signature generated with the current secret key and the index of the current interval can be verified by any user with the corresponding public key. A signature is verified as valid if the signature is a valid signature on the message under the given public key and the index of the interval included in the signature.

Forward secure signature schemes

The idea of forward security for digital signature schemes was introduced by Anderson [4] and later formalized in [8]. In one sentence, the forward security property says that even after a key compromise, all signatures created before remain valid.

A forward secure signature scheme (FSS) is a KES that provides the forward security property. The forward security property guarantees, that an adversary that is allowed to launch a chosen message attack for each interval and learns the secret key sk_i of an adaptively chosen interval i is unable to forge a signature for any interval $j < i$. In particular, this means that an adversary is unable to invert the key update algorithm KUpd.

To maintain the forward security in practice, it is indispensable that KUpd is executed on-time at the end of each interval and no keys of past intervals are kept e.g. as a back up. Otherwise, an adversary might be able to compromise a secret key sk_i of a past interval which would destroy the forward security.

For a formal definition of FSS we refer the reader to [8]. Note that forward security implies the standard notion of EU-CMA extended to the case of KES.

2.1.3 Public key encryption schemes

In public key encryption schemes, the key owner also has a key pair $(\mathsf{sk}, \mathsf{pk})$ which is generated with a key generation algorithm. The public key is used to encrypt messages, which can then only be decrypted with the corresponding private key. A public key encryption scheme is defined as follows.

A public key encryption scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $M \subseteq \{0,1\}^*$ is given by three algorithms:

KGen The key generation algorithm, on input of security parameter 1^λ, $\lambda \in \mathbb{N}$ generates a key pair $(\mathsf{sk}, \mathsf{pk})$ consisting of a secret decryption key sk and a public encryption key pk.

Enc The encryption algorithm, on input of a public key pk and a message $m \in M$, outputs a ciphertext c.

Dec The decryption algorithm, on input of a secret key sk and a ciphertext c outputs the decrypted message $m' \in \{invalid\} \cup M$

PKE is correct if for all $\lambda \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $m \in M$ and for $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ it is $m' = m$.

2.2 Public key infrastructures

A Public Key Infrastructure (PKI) supports the use of public key cryptography by handling keys and providing public key certificates, first introduced in [44]. The most basic operations of PKIs is the issuance and the revocation of certificates. Certificates bind the key owner's identity (e.g. a name) to his public key. In contrast, a revocation ends a previously established binding between an identity and a public key. The fundamental question is who is trusted to issue certificates. For this two basic principles exist. Either certificates are issued by users themselves to other users like in the OpenPGP Web of trust [75] or it is left to trusted third parties. The latter is the case in hierarchical PKIs, which are explained in the following. Whenever we refer to a PKI in this thesis, we refer to a hierarchical PKI.

2.2.1 Hierarchical PKI

In hierarchical PKIs, the trusted third party which is responsible for the issuance and management of certificates is called Certification Authority (CA). By issuing a certificate, a CA attests that a public key belongs to a particular entity, namely the subject of the certificate. Entities can request certificates from CAs. Before issuing a certificate, the CA must verify the identity of the requesting entity and that the public key for which the certificate is requested is indeed owned by the requesting entity, i.e. the requesting entity possesses the corresponding private key. As these verification processes are complex, a CA is usually supported by a Registration Authority (RA), which checks the credentials of a subject in order to attest its identity. A CA may in general be also supported by multiple RAs. A CA together with its RAs and further components is sometimes referred to as a Certification Service Provider [13]. Throughout this thesis we refer to entities authorized to issue certificates as CAs, disregarding its exact organizational structure.

The subject of a certificate may be an organization, an individual, a server or any other infrastructure component. The entity that actually contracts the CA and pays for the service of issuing certificates is called the subscriber. Subscriber may

be the subject of a certificate itself, or another entity. For example, an organization might request certificates for its employees.

Finally, there is the relying entity, which can be any individual or entity that relies on the certificates. For example, a relying entity may be an individual that verifies a signature or encrypts messages using a public key certified in a certificate. There is not necessarily a business relationship of any kind between a CA and the relying entity. However, a relying entity must trust the issuing CA in order to be convinced that the binding between subject and public key is valid.

In general, certificates are signed by the issuing CA (also referred to as the issuer). Thus, to verify the signature on a certificate, the relying entity must know the issuer's public key, and the relying entity must be able to verify the authenticity of that key.

In a hierarchical PKI, CAs are organized in a tree structure. The CA that builds the root of the tree is called Root CA. The Root CA (also called trust anchor) acts as the trust basis for the whole PKI. Root CAs sign certificates for subordinate CAs (Sub CAs) and end entities such as web servers or individuals. Sub CAs themselves sign certificates for other Sub CAs as well as end entities. A sample PKI is shown in Figure 2.1.

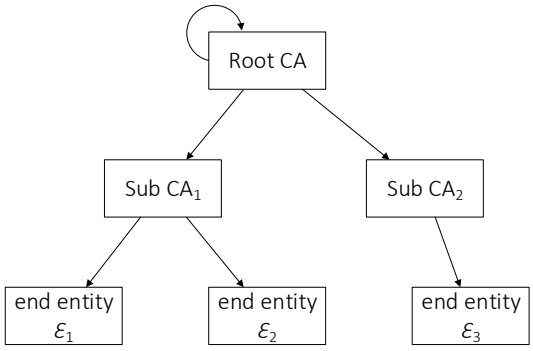


Figure 2.1: Hierarchical PKI. The boxes represent different entities, the arrows represent certificates.

By issuing a certificate to another CA, the issuer delegates trust to the Sub CA and authorizes the Sub CA to issue certificates itself. The authenticity of the Sub CA's key is guaranteed by the certificate issued by the CA on the next higher level. This reduces the key authentication problem to the secure pre-distribution of the Root CA's public key, which is to be realized via some out of band channel. Trust in the authenticity of an entity's public key is then established by the chain of certificates starting with the Root CA's certificate and ending with the end entity's certificate. This chain of certificates is called certification path. In a hierarchical PKI, a relying entity thus trusts any CA which has a valid certification path from

the Root CA to the CA's certificate.

The process of checking the certification path for correctness and validity is called path validation, which we explain in more detail in Section 2.2.4. In general, a certification path is a sequence $p = (C_1, ..., C_n)$ of $n \in \mathbb{N}$ certificates C_i, with the following properties [13]:

- Except for C_n, the subject of a certificate is the issuer of the subsequent certificate.
- The issuer of the first certificate C_1 is the trust anchor.
- The subject of the last certificate in the path C_n is the end entity.

The Root CA's certificate C_1 is usually a self-signed certificate. Self-signed means that a certificate is issued by an entity to itself and is signed with the private key associated to the public key in the certificate.

The most common type of hierarchical PKIs is the X.509 PKI which is an ITU-T standard [97] and was adopted by the Internet Engineering Task Force (IETF) [83]. In particular, the Web PKI which we explain in detail in Chapter 3 is an X.509 PKI. In the following, PKIs are assumed to be constructed according to the X.509 standard.

2.2.2 Certificates

As already stated, the binding between a public key and an identity is established through certificates. Certificates are data structures, that contain several fields. X.509 certificates are specified in the Abstract Syntax Notation version 1 (ASN.1) [96]. The most relevant contents of a certificate are the name of the subject and the subject's public key. Furthermore, a certificate specifies with which public key algorithm the certified public key is to be used. Also the name of the issuer is given in the certificate. This is required to identify the public key used to verify the issuer's signature on the certificate. Certificates also contain restrictions on the purpose for which the certified public key may be used. Even though a multitude of different key usages are possible, we only differentiate between CA certificates and end entity certificates. CA certificates are issued to (Sub) CAs and allow certificate signing. In general, this is not allowed for end entity certificates. To make certificates identifiable, they contain serial numbers. Together with the issuer's name, the serial number is a globally unique identifier for a certificate. Further auxiliary information can be included into certificates within so called certificate extensions. Please refer to [83] for details on standard extensions in X.509 certificates.

A certificate provides a binding between subject and public key only for a limited time period. This time period is called validity period and is defined in the certificate by the fields `NotBefore` and `NotAfter`. After the date given in the `NotAfter` field, a certificate is considered as expired. The binding between a public key and a subject can also be ended by a revocation, which we explain in the following section.

2.2.3 Revocation

Revocation means the invalidation of a certificate during its validity period. Revocation is done by the CA that certified the binding. The main mechanisms for revocation are Certificate Revocation Lists (CRL) [83] and the Online Certificate Status Protocol (OCSP) [104]. There are many reasons, why a revocation may be required, such as organizational changes, identity changes regarding the subject's identity, or the private key was compromised. In the last case, revocation is especially important, in order to prevent the malicious use of the key by an attacker. In order to trigger a revocation the compromise needs to be detected first. This is only possible with a certain delay. The time period between compromise and its detection is called gray period. During the gray period, the PKI is in an insecure state, in particular, when a CA's key is compromised.

CRL

CRLs are defined in [83]. CRLs are released by CAs in regular intervals and specify certificates that are revoked and have not yet expired. Within CRLs, the revoked certificates are identified by the issuer's name together with the certificate's serial number. Furthermore, they also may contain information about the reason for a certificate revocation and a date from which on the certificate needs to be considered invalid. There exist different forms of CRLs, such as full, delta or indirect CRLs which we do not explain here. For more details on the different CRL types we refer the reader to [13, 83].

CRLs are an offline mechanism for revocation checking. Relying entities can download the CRLs in regular time intervals from the CAs' servers and then use these CRLs to check the revocation status of certificates during path validation. From where to get the CRL relevant for a certain certificate is specified in the `cRLDistributionPoints` extension of the certificate.

OCSP

CRLs can get rather large and introduce delays into revocation until the next update of a CRL is published. Therefore, the Online Certificate Status Protocol [104] was defined. As the name says, it is an online mechanism to check the revocation status for a certificate. To provide the OCSP service, so called OCSP servers are operated, either by the CAs themselves or by an additional service provider. OCSP servers are special servers from which relying entities can request revocation status information of a certificate on demand. Within the request, the relying entity specifies the certificate using the issuer's name and the certificate's serial number. OCSP also allows to request the status of several certificates aggregated into one request.

The OCSP responder replies to a request with a signed response. Three different answers are possible:

- `revoked`, which states that the specified certificate has been revoked.
- `unknown`, which states that the OCSP server has no information about a certificate and is unable to give any answer about the status of the certificate.
- `good`, which indicates that the certificate is not revoked.

The answer `good` does not necessarily mean that a certificate is valid. The certificate might be expired or never have been issued. The actual behavior is left open to the implementation. OCSP servers that reply with `revoked` to requests for non-issued certificates must include the `extended-revoke` extension according to RFC 6960 [104].

Besides these possible status answers, an OCSP response may contain any information contained in CRLs, and includes the date showing when the response was generated.

OCSP is capable to provide up-to-date revocation information. However if OCSP responders base their answers on CRLs, this advantage is lost. To sign OCSP responses, OCSP servers normally use their own signing key and a certificate that was issued by the CA for which the OCSP server is operated. As for CRLs, the address of the OCSP server responsible for a certain certificate is specified in the `authorityInfoAccess` extension.

OCSP stapling OCSP stapling means that an OCSP response is not provided by the OCSP server to relying entities, but by the key owner himself. To do so, a key owner requests the status for his own certificate (or all certificates in the certification path of his certificate) from an OCSP server as described above. The responses are

then transferred to the relying entity by the key owner. As the responses are signed along with the date of their generation by the OCSP server, the relying entity can verify their authenticity and timeliness as in standard OCSP. This is especially relevant in scenarios where the key owner and the relying entity communicate directly and exchange the certification path during the communication as in TLS which will be detailed in Section 2.3.2. OCSP stapling has been specified as the certificate status request TLS extension [76] for a single certificate, and was extended to multiple certificates in [103].

Compared to the standard use of OCSP, OCSP stapling has the advantage that additional online communication with the OCSP server can be omitted. This saves overhead and also protects the relying entity's privacy, as the OCSP server is not able to track the communication of the relying entity. An additional benefit can be seen in the fact that the key owner can reuse the OCSP response for a certain time interval, depending on the required freshness of the response to be accepted by a relying entity. Thus, an OCSP response does not have to be newly generated every time a relying entity is verifies the key owner's certification path.

2.2.4 Path validation and validity models

In this section we explain how the validity of a certification path is evaluated. The outcome of path validation determines, whether a relying entity considers the public key certified within an end entity certificate as authentic or not.

There are two requirements for the validity of a certification path. The first is that for each digital signature on a certificate in the path, the corresponding verification algorithm Ver on input of the signature, the certificate (which is the signed message) and the issuer's public key must output 1, i.e., the signature is a valid signature under the given public key. This basic requirement is assumed to hold in the following.

The second requirement is the semantic correctness of the certification path. It is evaluated according to a validity model. The validity model specifies how the revocation information and the validity periods of the involved certificates are evaluated. In literature, three validity models can be found [6], which we shortly explain.

Shell model To formally describe the models let $n \in \mathbb{N}$ be the length of the certification path. C_1 is the self-signed certificate of the Root CA. C_n is the certificate of the end entity. We denote by $T_i(k)$ the starting date of the validity period of C_k and by $T_e(k)$ its expiration date. T_v denotes the time of verification. The shell model is defined as follows and depicted in Figure 2.2.

Definition 2.1 (Shell model)**.** *A certification path is valid at verification time T_v if all certificates in the certification path are valid at time T_v: $T_i(k) \leq T_v \leq T_e(k)$ for all $1 \leq k \leq n$ and no certificate is revoked at time T_v.*

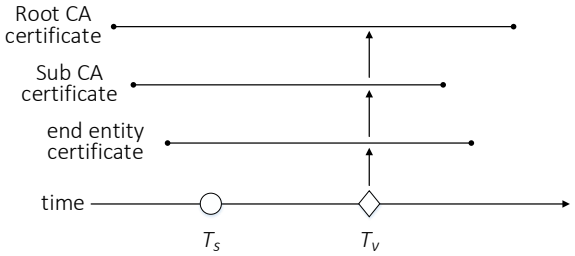


Figure 2.2: Shell model. The signature generation time is T_s and the verification time is T_v. Vertical arrows show the point in time used for validation of the certificates.

For a successful verification, all certificates in the path have to be valid at the time of verification. Thus, a relying entity can use the public key pk_n certified in C_n to encrypt messages or to verify signatures (depending on the allowed key usage) during the time period where all certificates are valid. Currently, most applications implement the shell model for certification path validation [6]. In particular, RFC 5280 [83] establishes the shell model as the standard validity model for the Web PKI.

Note that in the shell model a certification path becomes invalid once one of the certificates in the path expires or is revoked. This subsequently invalidates all signatures generated by subordinate entities. However, there are scenarios where this is not acceptable. For example, when data authenticity and non-repudiation are desired security goals. These properties must often be preserved for indefinite time periods. Therefore, two additional validity models have been defined for (end entity) signature validation. These models explicitly consider the points in time, where signatures have been generated.

Extended shell model In the following, let T_s be the time of signature generation by an end entity. Note that while the knowledge of T_v is trivial for a verifier (the relying entity), the knowledge of T_s is not, because T_s is not provided by a digital signature and thus requires a trustworthy time information, such as a time-stamp by a trusted third party (see Section 2.2.7). The end entity signature is also assumed to be a valid signature under the end entity's public key. Then, the extended shell model is defined as follows. Figure 2.3 depicts the model.

Definition 2.2 (Extended shell model)**.** *A digital signature is valid at verification time T_v if all certificates in the certification path are valid at time T_s: $T_i(k) \leq T_s \leq T_e(k)$ for all $1 \leq k \leq n$ and no certificate in the path is revoked at time T_s.*

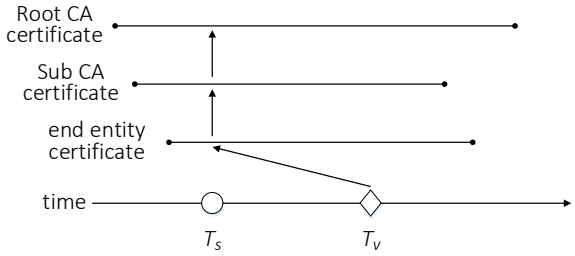


Figure 2.3: Extended shell model. The signature generation time is T_s and the verification time is T_v. Vertical arrows show the point in time used for validation of the certificates.

In the extended shell model (also called hybrid or modified shell model) T_s is used instead of T_v during validation. This means that the certificates in the path are checked for validity and revocation status at generation time of the end entity signature.

To implement this model, the signature generation time needs to be tied to the signature such that it can be checked that the certificates in the path were not revoked and were valid at that time. Furthermore, as the expiry of a certificate does not invalidate previously generated signatures, revocation information for the involved certificates must be preserved beyond certificate expirations, which in general is not necessary when applying the shell model. This also holds for the chain model explained in the following.

Chain Model The chain model is defined as follows and depicted in Figure 2.4.

Definition 2.3 (Chain model)**.** *A digital signature is valid at verification time T_v if:*

1. *The end entity certificate C_n is valid at the signing time T_s: $T_i(n) \leq T_s \leq T_e(n)$ and C_n is not revoked at time T_s.*

2. *Every CA certificate in the certification path is valid at the issuance time of the subordinate certificate in this path: $T_i(k-1) \leq T_i(k) \leq T_e(k-1)$ and the certificate C_{k-1} is not revoked at time $T_i(k)$ for all $2 \leq k \leq n$.*

In the chain model, any signature in the certification path is validated using its signature generation time, i.e. the issuance time of the certificate. The start date

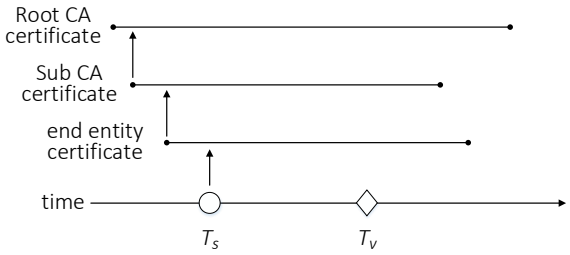


Figure 2.4: Chain model. The signature generation time is T_s and the verification time is T_v. Vertical arrows show the point in time used for validation of the certificates.

of the validity period can be used as an approximation. Thus, this date must lie within the validity period of the superordinate certificate.

If time-stamps are used in the implementation of the chain model, one time-stamp is required for every signature within the certification path [6]. This is because different dates prior to the signature generation time of the end entity signature have to be considered for the path validation. Both, the extended shell model as well as the chain model allow signature validation at any point in the future.

2.2.5 Certificate policies and certificate practice statements

In PKIs, the CAs are considered as trusted third parties. This trust is based on so called Certificate Policies (CP). The CP of a CA provides a set of rules for its operation and the internal procedures to which the CA committed itself. For example, a CP comprises information about authentication of subjects, security controls and also covers liability issues. A CA additionally describes the implementation of these rules in a Certificate Practice Statement (CPS). The CP and the CPS may be used by relying entities to evaluate whether to rely on the certificate issued by the CA in a specific application or not. A framework for CPs and CPSs is provided by [82].

2.2.6 Object identifiers

Object identifiers (OIDs) are used to refer to standardized objects, such as cryptographic algorithms or policies. OIDs can be used to identify any arbitrary object. For example, the OID 1.2.840.113549.1.1.1 identifies the RSA public key encryption scheme. OIDs are globally unique identifiers organized in a tree. The triadic root of the global OID tree is defined as follows:

0: ITU-T

1: ISO

2: joint-iso-itu-t

This means that the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) [93] is responsible for OIDs starting with 0, while the International Organization for Standardization (ISO) [92] is responsible for OIDs starting with 1. A leading 2 identifies jointly assigned OIDs.

2.2.7 Time-stamping

The process of adding a trusted time to electronic data is called time-stamping. Time-stamps are issued by a trusted third party called Time-Stamping Authority (TSA). Time-stamping can be implemented in different ways [77, 78, 95]. In general, time-stamping is achieved by signing the data along with the current date and time. With its signature, the TSA certifies that the time-stamped data existed at a certain point in time, typically at the time of the generation of the time-stamp. To verify a time-stamp, the TSA's public key must be known and authenticated. This in general is achieved with a hierarchical PKI.

Time-stamping is used for long term archiving of electronic documents and to achieve non-repudiation in practice [66]. Time-stamping can be provided as an additional service by a CA, which than acts as a TSA.

2.3 Internet communication

The Internet is a global open network of computer systems based on the TCP/IP protocol. It is subdivided into Autonomous Systems (AS) operated by different network operators called Internet Service Providers (ISP). An AS itself is a collection of IP networks. When data is sent from one computer system to another over the Internet, the data is routed through these different networks over a multitude of infrastructure devices such as routers. In general the route from the sender to the receiver is dynamically chosen depending on the configuration of the involved routers. In particular, the route is not chosen by the communication partners.

Web servers, to which we also refer as hosts, provide services to client systems. A typical example is accessing a web page on a web server through a web browser. The server is identified by its domain name given as a URL, such as `example.com`. The Domain Name System (DNS) [102] is responsible for resolving domain names to IP addresses, the actual network addresses where the server can be reached. To do so, DNS name servers are operated, for example by ISPs. From these name servers, clients can request the IP address for arbitrary domain names.

Basically, anyone can register for a domain name under which he can operate a web server. The entity who legitimately registered a domain is referred to as the domain owner.

This high level overview reveals that communication over the Internet in general is neither authentic nor confidential. Any device on the network path between communication partners can in principle act as a man-in-the-middle and alter and/or intercept the communication.

We first provide more information about DNS and DNSSEC. Afterwards, the Transport Layer Security (TLS) protocol is described, which provides secure communication.

2.3.1 DNS and DNSSEC

DNS is a hierarchical directory service for domain names. It divides the domain name space into so called zones according to the hierarchical structure of the domain name space. The zones are represented by data structures containing resource records. These resource records either point directly to IP addresses of hosts for specified domain names, or if not directly responsible for a domain name, to a subordinate DNS server, which maintains the respective child zone. The child zone in turn contains resource records, thus building a hierarchical structure.

DNS itself does not provide any protection mechanisms against the malicious alteration of DNS records. Therefore, DNS is susceptible to attacks like DNS spoofing or DNS cache poisoning, which aim at manipulating the IP address assigned to domain names, finally leading clients to a server which is actually not operated by the domain owner but by an attacker.

To counteract such attacks, the Domain Name System Security Extensions (DNSSEC) [79] have been specified and are currently in the process of being deployed. With DNSSEC, the zones and the contained records are digitally signed. The owners of the keys used for zone signing are the zone administrators. DNSSEC has a single trust anchor and each zone can only delegate trust to its direct child zones. Basically this is done by signing the hash of the key of the child zone.

2.3.2 Transport Layer Security

The TLS protocol [84] is the de facto standard for secure Internet communication.

It is the successor of SSL [88] which is considered insecure because of several vulnerabilities that have been detected in the past. TLS has been updated several times to its current version TLS 1.2. Many higher level communication protocols

such as HTTPS, FTPS or SMTPS build on TLS, which shows its outstanding importance for secure Internet communication. Through HTTPS, TLS secures the communication between web browsers and web servers.

TLS enables authentication and the establishment of confidential channels between clients and servers. Even though TLS supports client authentication, in most applications, e.g. in HTTPS, only web server authentication is used. Authentication is achieved using X.509 certificates. Domain owners subscribe at a CA for a certificate. The private key of their key pair is installed on the web server for authentication purposes. Thus, the security relies on the security of the underlying public key infrastructure, which is the Web PKI and will be described in Chapter 3.

During the TLS handshake, the server sends its certificate along with the certification path to the client (the relying entity). The client validates the certification path and if the validation is successful, client and server establish session keys. The key exchange protocol uses the server key pair. The server proves the possession of the private key and thus is authenticated as being the entity specified in the subject field of the certificate. The session keys are then used to encrypt the communication between client and server. This procedure ensures that the client communicates with the intended web server and that no unauthorized third party may access or manipulate the communication.

Browsers show the establishment of a secure channel to a web server by displaying locks and other visual items. If the validation of the server's certificate fails, warning messages are displayed to the user giving him a choice to abort the connection or to continue the potentially insecure communication.

2.4 Computational trust

In this thesis, we apply computational trust models to explicitly represent and evaluate trust placed in CAs. Computational trust is a means to support entities in making decisions under uncertainty, that is, under incomplete information. The two most widely-used definitions of trust are reliability trust and decision trust. Gambetta describes reliability trust as a subjective probability of performance without the option to monitor the performed action [20]. While this definition only captures the beliefs of an entity that is potentially unaffected by its trust, Jøsang defines decision trust as the will to depend on a trusted entity (Jøsang [40], inspired by McKnight and Chervany [55]):

Definition 2.4 (Decision trust)**.** *Decision trust is the extent to which a given party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.*

Computational trust models calculate an approximation to the quality of future interactions. Recommendations, experiences from previous interactions, and context-related indicators of trustworthiness can serve as input for this calculation. Based on the outcome, a decision can be made according to a pre-defined set of decision rules.

2.4.1 CertainTrust

The CertainTrust trust model by Ries [58] is used in this thesis. CertainTrust was extended with CertainLogic, which is a set of operators to combine CertainTrust opinions. These operators are similar to those of propositional logic, but consider the inherent uncertainty of CertainTrust opinions. CertainTrust and CertainLogic are equivalent to the Beta Reputation System and Subjective Logic, both by Jøsang et al. [37, 38, 39]. These models both rely on binary experiences that are combined using a Bayesian approach with beta probability density functions. An overview on different trust models that rely on this computational approach and similar ones can be found in the surveys by Jøsang et al. [40] and Ruohomaa et al. [60].

CertainTrust can handle and express trust-related information in two ways:

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

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

Between the experience space and the opinion space there exists an ambilateral mapping by parameterizing a Bayesian probability density function with the amount of positive and negative experiences. For details, see [57].

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

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

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

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

2.4.2 CertainLogic operators

There are several operators to combine different opinions. From two opinions about two independent statements a combined opinion about the statement regarding the truth of both input statements is computed with the AND operator of CertainLogic [58]:

Definition 2.5 (CertainLogic AND operator)**.** *Let A and B be independent statements and the opinions about these statements be given as $o_A = (t_A, c_A, f_A)$ and $o_B = (t_B, c_B, f_B)$. Then, the combined opinion on the statement regarding both A and B is defined as follows:*

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

The CertainLogic AND operator is commutative.

Besides combining independent opinions, there might be the need to combine dependent opinions, i.e. opinions on the same statement made by different entities. CertainLogic provides three FUSION operators for this task [28].

For this thesis, the cFUSION operator is required. It combines opinions by taking their inherent conflict into account. For example, asking different entities about the trustworthiness of a CA might result in two completely different opinions based on different previously made experiences. 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. [38, 28], do not account for conflict and only average the trust and certainty values of the resulting opinion. Furthermore, cFUSION allows to assign weights to input opinions to give them higher or lower importance. cFUSION is defined in [28]:

Definition 2.6 (CertainLogic cFUSION operator)**.** *Let A be a statement and let $o_{A_1} = (t_{A_1}, c_{A_1}, f_{A_1})$, $o_{A_2} = (t_{A_2}, c_{A_2}, f_{A_2})$, ..., $o_{A_n} = (t_{A_n}, c_{A_n}, f_{A_n})$ be n opinions associated to A. Furthermore, the weights w_1, w_2, ..., 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}, ..., o_{A_n}, respectively. The conflict-aware fusion of o_{A_1}, o_{A_2}, ..., o_{A_n} with degree of conflict DoC is denoted as:*

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

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

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

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

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

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

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

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

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

The CertainLogic cFUSION operator is commutative.

2.5 Further related work

In this section we shortly summarize further related work relevant in the context of this thesis.

The multitude of problems and disadvantages of the currently deployed Web PKI is described by well known researchers [18, 26, 27, 63]. Monitoring of the Web PKI reveals its enormous size and shows that indeed malpractices are common [17, 32, 125].

The enhancement of PKI with trust computation has previously been proposed by several researchers. However, the proposals so far lack practical applicability. Jøsang proposes an algebra for trust assessment in certification paths in [36] but mainly addresses trust networks similar to PGP [75]. Huang and Nicol [33] also define another trust model for trust assessment in PKI. Both approaches require trust values recommended by the intermediates to evaluate trust chains. Such recommendations are in general not included within commercial certificates. Different certificate classes like

domain validated (DV) or extended validation (EV) can be indicators for such trust values, but are to coarse grained for trust evaluation and are also not CA specific. Other researchers base trust evaluation in CAs on their policies and the adherence to those [15, 69]. This requires policy formalization [71, 15, 69] for automated processing. Such formalized policies are not provided by CAs, and are in general far to complex to be evaluated by the relying entities. Therefore, such approaches require technical and legal experts to process policies [70].

By including recommendations, systems using computational trust can generally be extended toward trust management systems. A survey on systems that evolve local trust into global trust can be found in [11] and, more specific to reputation-based trust management systems, in [60, 50, 31].

Key compromise and revocation can cause a huge impact on PKI systems, which is a well known problem. Many researchers have criticized how revocation is implemented in X.509. In this context, several proposals came up to either avoid revocation or mitigate its impact.

The complete elimination of revocation in PKIs by the use of short lived certificates is proposed by Rivest [59] and applied by e.g. Gassko et al.[21]. Yet, this approach comes with a considerable overhead of repeated certificate issuance and, in case of CA certificates, rebuilding the whole certificate hierarchy.

Other authors propose to distribute trust among multiple instances. While Maniatis et al. [22] propose a Byzantine-fault-tolerant network of TSAs to provide protection against TSA compromise, Tzvetkov [65] proposes a disaster coverable PKI model based on the majority trust principle. The first uses additional proofs of existence based on threshold signatures but requires a complex infrastructure and generates a huge overhead during verification of the signatures and time-stamps. In the latter, to tolerate the compromise of a minority of CAs, each certificate has to be signed in parallel by different CAs.

The use of write-once and widely witnessed media (e.g. official gazettes or newspapers) is an alternative to anchor digital objects in the time-line. Combined with the application of hash chains, as done by the TSA Surety [179], this can be implemented more efficient, but the usability and the preservation (e.g. of printed journals) in long term raise concerns.

Different proposals to use FSS in the area of PKI exist. Kim et al. [42] propose to use FSS for CAs to ensure business continuity in case of a CA key compromise. Thus, they use FSS in a similar way as this thesis. Yet, their work lacks the integration into the different PKI related mechanisms such as path validation in hierarchical PKIs and revocation. This is highly interwoven with the properties of FSS and must be adequately implemented to benefit from the specific properties.

Several other works apply FSS within PKI [24, 43, 72, 45, 46], but there are significant differences regarding the goals and the use of FSS compared to this thesis. Go [24] considers FSS for CAs, yet within the threshold setting in mobile ad-hoc networks – which significantly differs from our PKI setting – concluding that no existing scheme fulfills the specific requirements.

Koga et al. [43] propose a PKI model where the certificate chain for validation always has length one. They propose different constructions where either FSS or key-insulated signatures schemes (KIS) are used by the Root CA to generate the secret keys for the Sub CAs. While this allows the keys of Sub CAs to remain valid in case of a Root CA compromise, in case of the construction based on FSS, the compromise of a Sub CA implies the compromise of all Sub CAs that obtained keys with higher indices. Multiple Root CA key pairs or KIS solve this problem but at the cost of additional overhead. The KIS approach is further developed by Le et al. [45]. Nevertheless, in both works [43, 45] CAs always use their unique key to sign end entity certificates. Thus, all user certificates issued by a certain CA are invalidated in case of this CA being compromised. Furthermore, the CA keys need to be securely transferred from the Root to the Sub CAs. The approach of Le et al. even needs the transport of tamper resistant sub devices to the CAs.

In another work Le et al. [46] propose to use FSS in reverse order to allow to easily invalidate signed credentials. That is, a credential is signed with many keys obtained from an FSS key pair. Credentials can be invalidated by successively publishing the keys in reverse order. While this could be used to obtain short lived certificates, the applicability to establish a PKI is limited. As a key pair can obviously only be used to sign a single document, this would imply the management of a huge number of signing keys that are exposed to a possible compromise as the reverse order does not allow the deletion of former FSS keys.

Xu and Young [72] focus on compromise detection to finally obtain a robust system. To keep track of the key usage, signatures are deposited at highly secured systems and published on bulletin boards. FSS are used to provide a stateful authentication for the upload of signatures. Tampering can be observed based on inconsistencies in the authentication key states.

3 The Web PKI requires user-centric CA trust management

In this chapter we first give a problem exposition. Second, we show the practical relevance of the problems and third, it is shown that user-centric CA trust management is a promising solution to the described problems.

In Section 3.1 we show why the Web PKI – the globally trusted PKI, which is the basis for secure Internet communication – fails to provide trustworthy public keys. To show this, first the security model and the attacker model are presented. Based on these models, the Web PKI is analyzed and a detailed description of the problems leading to the failure of the Web PKI is presented.

The security model describes the involved entities and gives a definition of secure Internet communication which requires authenticity, integrity and confidentiality. In the attacker model, the attacker's goal, capabilities and limitations considered in this thesis are defined. Generally, the attackers we consider can generate certificates that are signed by a trusted CA and contain a subject chosen by the attacker. This capability can for example be achieved by compromising a CA's private key. Additionally, the considered attackers can inject certificates into TLS connections between web servers and clients, for example by DNS spoofing and acting as a man-in-the-middle on the network path between server and client. Based on the attacker and security model, we then show that the reasons for the failure of the Web PKI are the current trust model and shortcomings in the handling of revocation. The current trust model – which we refer to as the system-centric trust model – considers all public CAs fully trustworthy. The shortcomings in the handling of revocation refer to the reliable provision of revocation information and the hurdles preventing the revocation of CA keys, also known as the too-big-to-fail problem of CAs.

In Sections 3.2 and 3.3, we show the practical relevance of the trust management problem and the revocation problem described in the first part and that these are still open issues. This is done by analyzing publicly known security incidents and the examination of existing proposals that aim at the mitigation or the solution of

the problems of the Web PKI. We analyze these proposals regarding suitability and practicality.

In Section 3.4, it is shown that user-centric CA trust management, whose realization is presented in Chapters 4 and 5, is a promising solution for the trust management problem. The potential of user-centric CA trust management is evaluated by analyzing real world browsing histories and examining how relying entities experience the Web PKI. It is shown that relying entities individually only require a small subset of the globally trusted CAs and that user-centric CA trust management has the potential to reduce the attack surface spanned by the entirety of globally trusted CAs by more than 95%. Additionally, user-centric CA trust management enables the continuous monitoring of revocation information, which solves the problem of a reliable provision mechanism for revocation information. This will be shown in Chapter 4. For the too-big-to-fail problem, we provide a solution in Chapter 6. Section 3.5 concludes this chapter.

Parts of the contributions of this chapter were published in [B4], which covers the examination of the Web PKI from a user's perspective. This chapter extends the published results by a detailed analysis of the Web PKI and its weaknesses, the analysis of past CA security incidents as well as the evaluation of known concepts for the mitigation of the described weaknesses.

3.1 The defectiveness of the Web PKI

First the security and the attacker model are presented. Then, the Web PKI is analyzed and a detailed description of the problems causing the failure of the Web PKI is given.

3.1.1 Security model

In Chapters 3 - 6 of this thesis, secure communication over insecure networks, namely the Internet, is considered. Secure communication means, that a cryptographically secured connection is established between the communicating entities, where at least one of the entities authenticates itself to the communication partner. The secure connection provides authenticity, integrity and confidentiality. In general, this refers to secure communication between web browsers and web servers, where the web server authenticates itself to the web browser (and subsequently the Internet user that operates the browser). Secure connections are established using the TLS protocol. Authentication is achieved by presenting a public key certificate issued by a CA of the Web PKI. We give a formal description of the security model. The web

browser and the user operating it are considered as a unity and referred to as one entity.

In the security model we consider two entities $\mathcal{E}_1$ and $\mathcal{E}_2$. $\mathcal{E}_1$ establishes a TLS connection to $\mathcal{E}_2$ which needs to authenticate itself. The problem is to decide whether the connection is trustworthy for $\mathcal{E}_1$. A connection is considered trustworthy by $\mathcal{E}_1$, if $\mathcal{E}_2$'s public key pk that was used in the TLS connection establishment, is trusted by $\mathcal{E}_1$ to be a valid public key of $\mathcal{E}_2$. This requires:

1. A valid certificate C that binds pk to $\mathcal{E}_2$ is available to $\mathcal{E}_1$.
2. $\mathcal{E}_1$ trusts the issuer of C.

The first requirement is a standard PKI issue (cf. Chapter 2.2 for details). To fulfill requirement 1, $\mathcal{E}_1$ requires a certification path $p = (C_1, ..., C_n)$ such that:

(a) $C_n = C$

(b) p passes path validation

Requirement 2 is fulfilled if p additionally passes *trust validation*. Explicit trust validation is not incorporated in the current deployment of the Web PKI but trustworthiness is assumed for all CAs of the Web PKI (cf. Section 3.1.3). Explicit trust validation is the core mechanism of user-centric trust management, which is described in Chapters 4 and 5.

Note that a trustworthy connection to $\mathcal{E}_2$ does not imply the trustworthiness of $\mathcal{E}_2$ itself. The trustworthiness of $\mathcal{E}_2$ comprises, for example, the quality of its web page and the provided content. This is not addressed in this thesis and requires additional mechanisms like the Web of Trust [190] or commercial web page ratings, e.g., Norton SafeWeb [166] or McAfee SiteAdvisor [146]. In general, authentication is a basic requirement for such mechanisms.

3.1.2 Attacker model

In the following, the attacker's goal, capabilities, and limitations are defined. The attacker $\mathcal{A}$ aims at breaking the authenticity, integrity, and/or confidentiality of secure communication which was described in the previous section. The availability of communication is out of scope of this thesis. We focus on attackers that make use of the malfunctioning of the Web PKI to attack the communication. In Section 3.1.3, we explain why the Web PKI fails to prevent such attacks. In general, the TLS protocol and the used cryptographic primitives are assumed to be secure. TLS vulnerabilities, as well as implementation errors are out of scope of this thesis.

Attacker goal $\mathcal{A}$ targets at TLS connections of a relying entity $\mathcal{E}_1$. The goal is to impersonate web server $\mathcal{E}_2$, which is the intended communication partner. During the attack, $\mathcal{A}$ may be the end point of the communication channel or acts as a man-in-the-middle in the secure communication as shown in Figure 3.1. The attack is successful if $\mathcal{A}$ is not detected. To achieve this, $\mathcal{A}$ presents a fraudulent certificate to $\mathcal{E}_1$ for which he controls the private key during TLS connection establishment. This allows $\mathcal{A}$ to impersonate $\mathcal{E}_2$ towards $\mathcal{E}_1$. $\mathcal{A}$ optionally establishes a second connection to $\mathcal{E}_2$. This gives $\mathcal{A}$ full control over the communication, while the attack is transparent to $\mathcal{E}_1$ and $\mathcal{E}_2$.

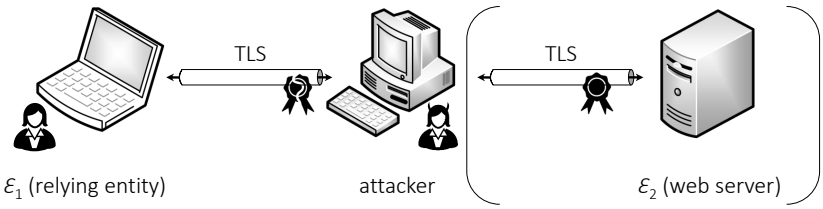


Figure 3.1: Attacker impersonating the intended communication partner.

Attacker capabilities $\mathcal{A}$ is generally capable to position himself on the network path between $\mathcal{E}_1$ and $\mathcal{E}_2$, for example by DNS spoofing. Additionally, $\mathcal{A}$ can generate certificates that are signed by a CA of the Web PKI and contain a subject chosen by $\mathcal{A}$. We refer to such certificates as fraudulent certificates. In contrast, certificates where the subject actually identifies the entity who controls the corresponding private key are called legitimate certificates. Fraudulent certificates may be issued by a CA for example by compromising a CA's private key and using it to issue certificates. Note that self-signed certificates are unsuitable for attacks as they will be detected during standard path validation.

Attacker limitations $\mathcal{A}$ can only generate fraudulent certificates on behalf of one CA of the Web PKI at the same time. The CA is chosen by $\mathcal{A}$ and fixed afterward. This assumption is justified by the fact that the issuance of fraudulent certificates on behalf of a CA is not trivial in practice. We discuss security incidents that enabled an attacker to manage such a malicious issuance in Section 3.2.

$\mathcal{E}_1$ and $\mathcal{E}_2$'s IT-systems are assumed to be secure and not compromised. Thus, $\mathcal{A}$ is unable to access or manipulate locally stored data. Furthermore, $\mathcal{A}$ has no access to $\mathcal{E}_2$'s private key corresponding to $\mathcal{E}_2$'s public key pk certified in $\mathcal{E}_2$'s legitimate certificate. $\mathcal{A}$ is not capable of breaking the cryptographic algorithms or circumvent the establishment of secure connections. In particular this means, once a secure

channel has been established between $\mathcal{E}_1$ and $\mathcal{E}_2$ using $\mathcal{E}_2$'s legitimate certificate, $\mathcal{A}$ cannot eavesdrop on the content of the communication or manipulate the data without being detected.

3.1.3 The Web PKI

With the success of the Internet, the Web PKI gained more and more importance. Its size and its complexity have continuously been growing. In recent years severe vulnerabilities have been discovered in the design and implementation of the Web PKI. They have led to the ongoing crisis of confidence in the Web PKI. According to the IETF Web PKI OPS Working Group [107], the Web PKI is defined as follows:

Definition 3.1 (Web PKI)**.** *The Web Public Key Infrastructure (Web PKI) is the set of systems, policies, procedures and people required to issue manage, distribute, use, store and revoke public key certificates in order to protect the confidentiality, integrity, and authenticity of communications between Web browsers and Web content servers.*

In this thesis we follow this definition. The CAs of the Web PKI are the globally trusted CAs that issue certificates to web content and application providers. The use of the certificates issued by the Web PKI for the TLS protocols makes the Web PKI to the indispensable basis for secure Internet communication.

However, the Web PKI fails in many respects to provide the desired security. This is described in the following. We start with an overview on the Web PKI and a description of its characteristics. Then, the current trust model – which we refer to as the system-centric trust model – and the shortcomings regarding revocation handling as the reasons for the failure of the Web PKI are discussed in detail.

The characteristics of the Web PKI

The Web PKI is a hierarchical PKI according to the X.509 standard [83]. Special about the Web PKI is its global nature. Because of its scope to enable secure communication on the Internet, interoperability is a central requirement. On the one hand, the group of relying entities is virtually unlimited. Namely, any of the more than 3 billion (stats for June 30, 2014 [135]) individual Internet users as well as organizations and governments rely on certificates issued by the Web PKI. In general, the relying entities are non-experts and cannot be expected to understand the functioning of a PKI [5]. On the other hand, the group of potential certificate subjects is virtually unlimited. Any web server operator or domain owner may own

one or more certificates. The customers of CAs can be organizations, governments, or individuals.

To deal with this complexity, the Web PKI has evolved to a global system of CAs distributed around the world. These CAs are not organized in a strict hierarchy with a single Root CA, but a set of Root CAs which serve as trust anchors for certificate validation. Each of the Root CAs spans a hierarchical PKI as explained in Section 2.2, while it is also possible that Sub CAs have their keys certified by more than one superordinate CAs. Because of the huge number of CAs and the fact that relying entities are non-experts, it is impossible to leave the management of trust anchors to the relying entities. Instead, this is done by browser and operating system vendors. The public keys of Root CAs are distributed within trusted lists called *root stores*, contained in operating systems and browsers. Thus, browser and operating system vendors globally define – according to their specific policies [148, 157] which comprise certain security and audit requirements – which CAs are trusted. Besides these individual rule sets, the CA/Browser Forum has defined baseline requirements [81] that CAs of the Web PKI must meet and reflect in their policies. For Root CAs, the adherence to the different policies is enforced through annual security audits, while it is in general left to the CAs themselves to ensure this for Sub CAs.

The number of Root CAs has been constantly growing. For example, the root store of the Mozilla browser comes together with the NSS crypto library [155] and contains about 160 CAs [32, 156] while Microsoft's root store even contains about 264 CAs [63]. The total number of trusted CAs can only be estimated through broad scale web scanning surveys [17, 32, 125] because there is no public directory that identifies the existing (Sub) CAs. In this thesis, we use the numbers presented by Durumeric et al. [17], which is the most complete survey to the best of our knowledge. By scanning the address space of the Internet and downloading the certification paths from publicly visible web servers, the CAs of the Web PKI are identified. They are extracted from the collected certification paths by the use of their issuer names in conjunction with their public keys. This results in a lower bound of 1,590 trusted CAs which are controlled by 683 private as well as governmental organizations and are operated under the jurisdictions of 57 different countries.

Also, there is no central directory service where end entity certificates are registered. This also results from the global nature of the Web PKI, where a central directory does not scale. In general, a web server presents its certificate along with the certification path to the relying entity during the TLS handshake. The relying entity validates the certification path and checks if it starts with a trusted Root CA and if the last certificate identifies the intended communication partner. If so the relying entity trusts in the authenticity of the server. The public key is extracted

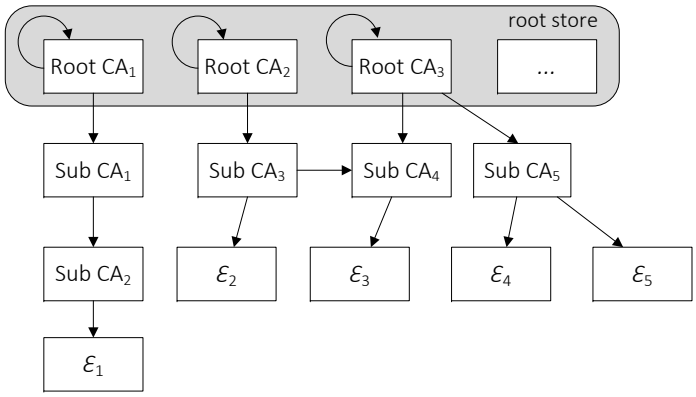


Figure 3.2: Example Web PKI

from the certificate and used to establish session keys to secure the communication.

A simplified example of the resulting Web PKI is depicted in Fig. 3.2. Here, an exemplary certification path exists from Root CA_1 to end entity $\mathcal{E}_1$, where the arrows represent certificates. The circular arrows represent self-signed certificates, which are often issued by Root CAs to themselves in order to publish their keys. To validate $\mathcal{E}_1$'s certificate, only the key of Root CA_1 must be known. All other keys are contained in the intermediary certificates. In this small example, it can also be seen, that it is difficult to determine all the trusted CAs. For example if Root CA_3 were removed from the root store, its direct Sub CA_4 would still be trusted due to the additional chain from Root CA_2. However, as there exists no public directory of all certificates, the existing chains are in general unknown to relying entities until they are presented during connection establishment.

Having explained the set up of the Web PKI, in the following sections the main problems – the system-centric trust model and the shortcomings in the handling of revocation – are described and it is shown, how they support the attackers described in Section 3.1.2.

The system-centric trust model of the Web PKI

The universal applicability of the Web PKI makes it impossible to differentiate between CAs on a global level. Any restriction on the trustworthiness of CAs by browsers and operating systems also limits their operational capability which contradicts the requirement of universal applicability, i.e., browsers and operating systems must work for anybody and thus need to be capable of verifying any certificate deployed on web servers.

This constitutes the system-centric trust model. By this we mean that Root

CAs as well as Sub CAs are generally considered as trusted third parties, i.e. they are assumed to be fully trustworthy. By issuing a Sub CA's certificate, the issuer delegates all its privileges to the Sub CA, which leads to the above mentioned 1,590 fully trusted CAs. Also, there are no restrictions for CAs with respect to the domain names for which they are allowed to issue certificates or certificate uses. Although the name constraints extension [83] can be used to limit the power of a CA, in practice it is almost never used [1]. Thus, each of the trusted CAs can sign certificates for any web service or domain.

The system-centric trust model directly leads to a *weakest-link* situation, where the security of the whole system is determined by the weakest CA. An attacker as defined in Section 3.1.2 who can obtain fraudulent certificates (by compromising a CA or by a CA failure) from one of the trusted CAs can potentially impersonate any web server and mount a man-in-the-middle attack on any TLS secured connection without users even noticing the attack. Thus, the system-centric trust model creates an attack surface growing with each additional trusted CA. This attack surface determines the attacker's capability concerning the acquisition of a fraudulent certificate.

The baseline requirements, certificate policies, and security audits enforce a basic level of security. However, they cannot provide guarantees. The security incidents in the past clearly show that the issuance and use of fraudulent certificates is a real threat (cf. Section 3.2). It has also been shown [23] that certificate policies do not allow to differentiate between CAs regarding their security in a fine grained manner.

As it is impossible to completely eliminate CA failures, the goal must be to minimize the attack surface of the Web PKI. However, as a global limitation of the trusted CAs is not possible, we propose user-centric CA trust management, where trust decisions are made on a per user level. In Section 3.4 the potential of the approach is shown. Its realization is presented in Chapters 4 and 5.

Revocation in the Web PKI

Revocation of a certificate is required whenever the certified key needs to be invalidated before the end of its validity period. Revocation is indispensable as it transitions the system into a secure state after key compromise, or a malicious or erroneous certificate issuance. Without revocation, an attacker can use fraudulent certificates until they expire, which in general is an intolerable time span. Given a revocation of a certificate, the attacker cannot further use it for attacks as soon as the relying entity is aware of the revocation.

However, the Web PKI faces several problems concerning the revocation of certifi-

cates: the reliable provision of revocation information and the *too-big-to-fail* problem.

First we describe the provision of revocation information. A relying entity must obtain the revocation information during TLS connection establishment after having obtained the certification path of the server's certificate. Ideally, the connection establishment and page loading must be blocked until the revocation status of the certificate in question is checked. However, these online revocation checks on the one hand introduce latency, which users are not willing to accept [56] and on the other hand, there is no guarantee that the CA's OCSP or CRL service is accessible at the time when revocation checking is required. Because these online revocation checks fail so often, all major browser vendors turned to so called soft-fails[1] [182]. This means that if the revocation information for a certificate cannot be obtained, it is simply evaluated as not revoked. However, this renders revocation useless exactly when it is required, namely in the presence of an attacker as defined in Section 3.1.2. The attacker is able to manipulate the relying entity's communication. Thus, he can also block revocation checking leading to the acceptance of the fraudulent certificate, although it might have already been revoked [138, 159]. OCSP stapling does not solve this problem, as again the attacker can suppress the OCSP response (cf. Section 3.3). In Chapter 4 it is shown how user-centric CA trust management solves the provision problem by enabling the continuous monitoring of revocation information.

The too-big-to-fail problem of CAs refers to the practical impossibility to revoke CA keys. It is more severe the more services depend on the respective CA and the more critical these services are. The revocation of a CA certificate subsequently invalidates all certificates that contain the revoked certificate in their certification path because of the validation according to the shell model (cf. Definition 2.1). This in turn means, that all services using such an invalidated certificate for authentication become unavailable until their certificates are exchanged by new ones. Considering the huge customer bases of CAs of the Web PKI shows that deploying new certificates on all affected web servers is problematic. This is not an automated process but requires manual changes by web server operators. In many scenarios, such a temporal unavailability of services is not acceptable and thus, either requires that the revocation of a CA key is considerably delayed or even completely avoided. For example, revoking a certain Comodo Root CA would invalidate more than 200,000 TLS certificates [5] making the corresponding web services unavailable. The publicly known incidents evaluated in Section 3.2 show the relevance of this problem. In none

[1] Note that Google even disabled online revocation checking completely in the Chrome browser [141].

of the incidents, browser or operating system vendors were willing to let security go beyond connectivity. Furthermore, the huge impacts of a CA certificate revocation and the related effects for the CA's reputation provide strong incentives not to report security breaches in order to circumvent public attention [5]. A solution for the too-big-to-fail problem is provided in Chapter 6.

3.2 CA failures and compromises

We show the practical relevance of the trust management and the revocation problems described in Section 3.1.3. This is done by analyzing publicly known security incidents concerning the CAs of the Web PKI.

Security incidents have several different reasons. On the one hand, a CA may intentionally misbehave, e.g., due to economic motives, governmental orders, or a CA is owned by the government and is used for surveillance [63]. In such cases the CA intentionally issues fraudulent certificates to be used to attack TLS secured connections. Subsequently, no security mechanisms set up by the CA itself can help to detect or prevent such attacks. On the other hand, security incidents may result from unintentional misbehavior, e.g., because of weak security practices or implementation errors, social engineering and other attack vectors that are exploited by attackers to obtain fraudulent certificates.

3.2.1 Categories of CA security incidents

The unintentional security incidents can further be categorized into four scenarios. These four scenarios are the following [184]:

- Impersonation: The attacker impersonates another entity when registering with a CA and tricks the RA into falsely validating the attacker's pretended identity. This leads to the issuance of a certificate binding the attacker's key to the identity of the impersonated entity.

- RA compromise: The second scenario goes one step further. In this scenario, the attacker infiltrates the RA systems, e.g., by malware or stealing the credentials of a legitimate RA user account and is enabled to directly authorize the issuance of (arbitrary) certificates by the related CA.

- CA system compromise: The CA system compromise describes the scenario, where the attacker directly manipulates the certificate signing processes. In this scenario, the attacker does not have access to CA's signing key, but is able

to directly initiate the issuance of fraudulent certificates, e.g., by querying the security module containing the key. As the attacker has access to the CA systems, he may also be able to manipulate security mechanisms of the CA such as the logging mechanisms and thus conceal the compromise.

- CA signing key compromise: This is the most severe scenario. In this case, the attacker obtains a copy of the CA's private key, e.g., by stealing the key, attacking the underlying algorithm or by a cryptanalytic attack. With the CA's key the attacker can arbitrarily issue fraudulent certificates, even after he loses control over the compromised CA systems.

3.2.2 History of CA security incidents

In this section, publicly known CA security incidents are evaluated. Regardless the severity of the failures and the threats for global communication no official investigation and reporting from the affected CA's side about the incidents can be found in many cases. The sources of information are discussions between security experts found on their Internet blogs, discussion forums of browser and operating system vendors, or the online press.

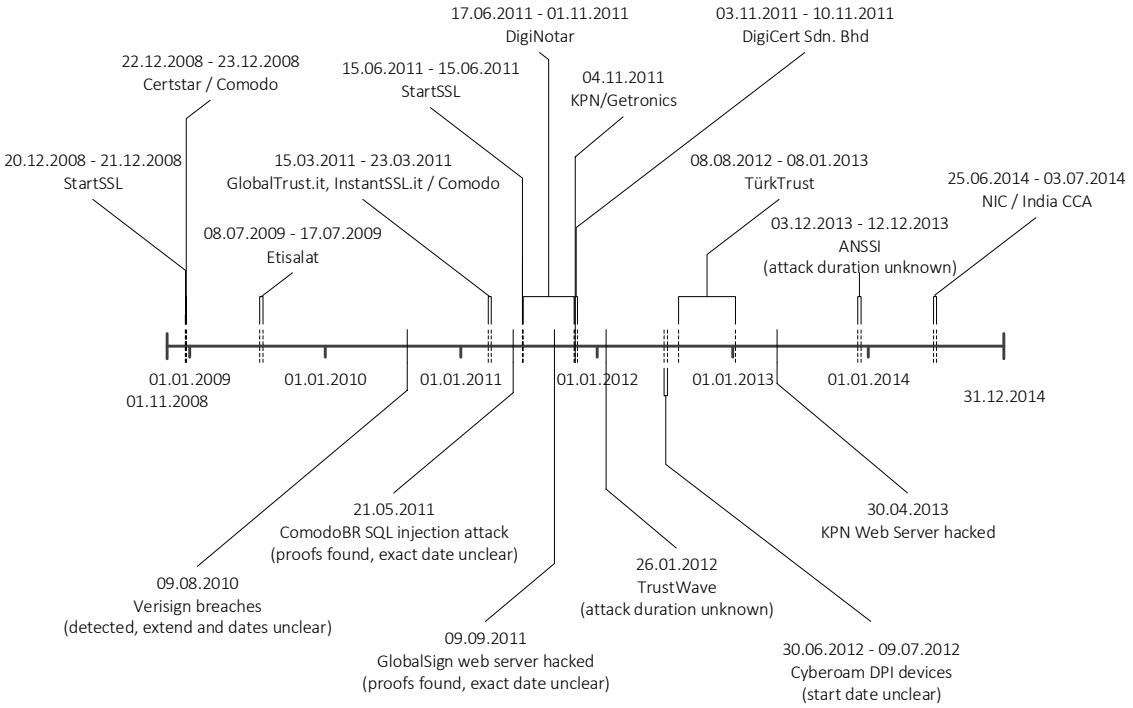


Figure 3.3: Timeline of CA failures, compromises and attacks

Figure 3.3 shows the timeline of publicly disclosed security incidents. In the following we evaluate these incidents and the respective countermeasures that were taken upon detection. This will show, that the system-centric trust model is not justified, the reactions on security incidents are highly influenced by the too-big-to-fail problem and furthermore, revocation as currently implemented within the Web PKI is not at all trusted to fulfill its purpose. We note that it is a general presumption that the estimated number of unreported cases is much higher than the publicly disclosed incidents [5, 114, 188, 27].

Impersonation

There are many ways how impersonation attacks can be carried out depending on the mechanisms in place to verify the subscriber's identity. Examples are using a compromised employee email account, faxing forged business licenses and even a case, where the attacker simply told the CA not to do anything bad with the certificate has been reported [27]. Most of those incidents do not lead to a public disclosure but to a silent revocation if the erroneous issuance is detected. Even an incident in 2001 where Verisign erroneously issued two code signing certificates for Microsoft products to an attacker only lead to a short note on the Microsoft support pages [153]. The certificates did not contain a CRL distribution point extension and thus a Windows update was required in order to invalidate them. In the following, we discuss two impersonation incidents which have been covered by the media and lead to broad discussions in the community.

Both incidents happened by the end of 2008, one at StartCom ltd., which maintains the StartSSL Root CA and the other one at CertStar, one of Comodo's resellers of certificates issued by PositiveSSL CA, a Comodo Root CA. While both vulnerabilities in the verification processes where discovered by security experts, there are many differences in the two incidents. The one at StartCom resulted from an implementation error. The exploit was detected by unspecified security mechanisms implemented by the CA while the exploit was still running and the erroneously issued certificates were immediately revoked (within a few minutes) by publishing a CRL. Furthermore, investigations immediately started and lead to an update of the CA's systems as well as a full disclosure of the incident by the CA itself [163, 164]. In contrast, the impersonation attack at CertStar was possible because of the complete absence of the verification of the subscriber's identity. In order to prove this, the attacker bought a certificate for mozilla.com and publicly disclosed the vulnerability [112, 110]. Only because of the disclosure by the attacker, the vulnerability was noticed. Comodo revoked the mozilla.com certificate and initiated an audit of

CertStar's RA processes as well as suspended the RA's ability to request certificates during the investigation. However, not only a report of the investigation results is still missing, but Comodo also played down the incident and claimed that the vulnerability is simply a well known problem of domain validated certificates [110]. Furthermore, because of to the absence of control mechanisms, it is totally unclear if the vulnerability was previously exploited by real attackers.

The incidents illustrate that the system-centric trust model is not justified.

RA compromises

In 2011, severe RA compromises happened at two of Comodo's resellers GlobalTrust.it and InstantSSL.it [144]. Both had the privileges to request certificates from the Comodo owned CA UTN-USERFirst-Hardware [122] without any further approvals. This practice makes an RA compromise nearly as severe as a CA system compromise, with the difference that attackers are not able to manipulate the CA's monitoring and logging systems. The attacker managed to compromise user accounts of the RA and requested 9 certificates for high value domains, of which at least one was used during man-in-the-middle attacks in Iran [122, 117]. The attack was detected within a time span of several hours and the certificates where revoked. This time, Comodo additionally informed all major browser vendors, the related domain owners as well as governmental authorities. Subsequently, the browser vendors additionally blacklisted the fraudulent certificates within their browsers [137, 185, 109, 152]. It took one week until all browsers where patched. This was the first time, that such black listing functionality was used to deal with fraudulent certificates. A public disclosure in the press was actively hold back until all updates where finished [109]. In the aftermath, Comodo investigated the incident and published an incident report [117]. Additional control and detection mechanisms where set up which turned out to be effective during another attack on RA systems one week later, where the issuance of certificates could be prevented. Besides that, the organizational structure was changed such that each RA obtains its own Sub CA certificate [108].

First, the incident shows that revocation cannot be considered to securely prevent attacks with fraudulent certificates. Second, it shows the relevance of the too-big-to-fail problem as the certificate of the issuing CA could not be revoked due to its relevance in the certification business [27], which ultimately lead to the change in Comodo's business practices. Finally, it demonstrates the tendency to prevent publication as described in [5], which leaves relying entities unprotected until the final elimination of the threat.

CA system compromises

In 2011, two CA system compromises occurred, which are both assumed to be related to the Comodo RA compromise earlier that year. One occurred at the StartSSL CA and one at DigiNotar. The attack on StartSSL happened in June 2011 and the attacker gained access to the CA systems. Yet, the attack was detected in real time due to monitoring and other unspecified security mechanisms [174] and the issuance of certificates could be prevented.

The DigiNotar compromise, which presumably was initiated by the Iranian government, turned out to be the most severe incident related to certification that happened in the history of the Web PKI. The severity of the incident, "which put the security and privacy of millions of citizens at risk" [127] and threatened the lives of people in Iran, was due to an accumulation of missing or inappropriate security mechanisms combined with misjudgment, inadequacies in reporting about the incident to relying entities and governments and the attempt to conceal the whole incident from the public. This had severe consequences and exposed at least 300,000 Iranian Internet users to man-in-the-middle attacks and the interception of personal communication. It was even argued, that activists had died as a consequence of this security breach [127]. The attack(s) took place over the period of nearly one month until they were detected. During this time, the attacker successively compromised the CA systems until he obtained administrative access to all of DigiNotar's CA servers including the qualified ones managing the Dutch governmental PKIOverheid. However, the attack was not publicly disclosed upon detection. This happened an additional month later, when fraudulent certificates for Google were detected by Chrome users because of Google's pinning mechanism for Google pages. At that time, DigiNotar informed the Dutch government as well as major browser vendors. Yet, in contrast to the Comodo RA compromise, the fraudulent certificates could not be blacklisted as they were unknown because the attackers had also manipulated the CA log files. This finally lead to the decision to completely revoke all DigiNotar CA certificates and remove it as trusted CA from the Web PKI, which ultimately lead to the bankruptcy of DigiNotar. However, the final and complete removal was delayed for two more months. The reason for this intended delay was not to interrupt the proper functioning of governmental applications that widely relied on DigiNotar certificates. Investigations of the external IT specialist FoxIT later revealed that a minimum of 531 fraudulent certificates for high value domains like Google, Facebook, governmental pages as well as CA certificates on the name of several other CAs of the Web PKI were issued [129].

Another CA system compromise happened mid 2014 [140] at the National Infor-

matics Centre (NIC) which is a Sub CA of the India CCA Root CA in the Microsoft root store. Again it was detected by Google's pinning mechanism and not by the CA itself. Thus, the exact duration of the attack is unknown. Google blacklisted the certificates and alerted NIC, India CCA and Microsoft. This time, all CA certificates of NIC could be revoked within one day since NIC had only limited relevance in the certification business. Investigations by India CCA revealed that NIC's issuance process was compromised and certificates for Google and Yahoo were issued to the attacker. However, Google detected additional certificates other than those reported by India CCA. This puts the reliability of the investigations in question and shows that the complete extent of the incident is unknown.

These incidents again illustrate that the system-centric trust model is inappropriate and reveal the threats that can result from fraudulent certificates. Besides that, the DigiNotar incident shows that the too-big-to-fail problem also concerns CAs that have a relatively low relevance compared to the big players in the certification business. However, work arounds that avoid the revocation of CA certificates are not always effective which shows the necessity of the solution for the too-big-to-fail problem.

CA signing key compromises

Regarding CA signing key compromises no incidents have been reported so far. Given such an incident, there is no other measure than revoking the certificate that certifies the compromised key. This is because in such a case, an attacker can issue arbitrary certificates with that key, without any possibility to stop the attacker from doing this.

CA misbehavior

Intentional CA misbehavior is another problem in the Web PKI. Also incidents where a governmental order forces a CA to issue fraudulent certificates, see e.g. [63], belong to this class if incidents. The special characteristic of such incidents is that the CA has full control over its private signing key. No internal control mechanisms can prevent such misbehavior because no real attack on the CA systems takes place.

Over the years a multitude of such incidents where revealed, which illustrate the inadequacy of the system-centric trust model. The first one which has been publicly reported happened in 2009, when Etisalat, a telecommunication company of the United Arab Emirates, misused a code signing certificate it legitimately owned to distribute spyware to the Blackberry subscribers among its customers [186, 172]. The spyware was masked as a regular update by RIM [173]. The incident was only

detected by chance, as the spyware lead to a significant loss in battery life. The Sub CA certificate also owned by Etisalat was – as far it is known – not misused, and therefore was never revoked [120]. Thus, Etisalate is still a trusted CA of the Web PKI even though it may be suspected that a company which intentionally enabled the surveillance of 145,000 customers [143], also makes malicious use of its CA certificate.

In 2011, the Malaysian company DigiCert Sdn. Bhd acted not really malicious but highly irresponsible and violated the policies and baseline requirements of the CA/Browser Forum [81]. DigiCert Sdn. Bhd issued 22 certificates certifying weak 512 bit RSA keys, which are insecure and prone to factorization. Furthermore, strictly required extensions as CRL distribution points were missing [126, 136]. This made the revocation of the affected certificates useless. Entrust, the parental CA detected these policy violating practices and informed the major browser vendors. Subsequently, DigiCert's CA certificate was revoked by Entrust and all major browser vendors blacklisted the certificates in question within one week after detection [165, 116, 149].

In 2012, a policy update of Trustwave revealed that prior to the update Trustwave knowingly sold CA certificates to its organizational customers for monitoring purposes within organizational networks. This allowed a hidden surveillance of the employees (and potentially of any other Internet user) as there is no need to manually install an additional trusted CA certificate on the employees' systems as normally required when such monitoring devices are used. In a clarification statement Trustwave notes [183] that this was only done once and the certificate was installed within a hardware security module to prevent misuse of the certificate outside the organization. But, Trustwave claims that this is common practice in the business [5]. Such behavior becomes even more questionable, when taking into account another incident at Cyberoam in 2012. It turned out that the deep packet inspection devices sold by Cyberoam [118] allowed the export of the installed CA key [113]. Cyberoam itself is not a CA of the Web PKI, however the combination of these two incidents shows the threat potential of Trustwaves former business practices.

Also in 2012, TürkTrust erroneously issued CA certificates instead of end entity certificates to two of its customers. One of these certificates was revoked two days later on request of the customer who detected the error. The incident became public more then four month later, when the second CA certificate which was still not revoked, was used in a man-in-the-middle attack. The attack was again detected by the pinning mechanism of the Chrome browser. Investigations by TürkTrust revealed that the erroneously issued Sub CA certificate was installed on a company's checkpoint firewall [111] to regulate access to Google by internal clients. By then,

TürkTrust also revoked the second fraudulent certificate. However, the fact that TürkTrust was not aware of the second erroneously issued CA certificate illustrates its insufficient security practices. Even though no signs of exploits for fraudulent purposes [176] could be found, the erroneously issued certificates were additionally blacklisted in major browsers [150, 176].

At the end of 2013, it turned out, that a certificate which chained up to a public Root CA of the Web PKI, namely ANSSI a CA of the French government, was used to inspect encrypted traffic of employees [139]. Subsequently, ANSSI revoked the Sub CA certificate which was used to issue the certificate employed during the monitoring activities. Additionally, the Sub CA certificate was blacklisted within browsers [189, 151] while Google additionally limited trust for ANSSI to only issue certificates for French and related top level domains [139]. While the impact of this incident is assumed to be limited [139], it is another example for untrustworthy behavior of CAs and policy violations.

Further incidents

Several other security incidents at CAs but not directly related to the certificate issuing systems happened over the past years. In 2012, an investigation by Reuters [147] revealed that the corporate network of Verisign was breached back in 2010. Verisign never reported this incident except for a short comment in a in quarterly report in 2011 [187] which does not reveal details, thus leaving the extent of the breach totally unclear. In 2011, another Comodo RA, namely ComodoBR [132] was target of a SQL-injection attack on the RA's web facing servers, that revealed information related to certificate signing requests, in addition to email addresses, user IDs, and password information for a limited number of employees but did not enable the attacker to make use of the certification infrastructure. Also in 2011, a peripheral web server of GlobalSign was hacked [131] which lead to the temporal suspension of certificate issuance and further investigations. These two incidents are commonly assumed to be related to the other RA and CA compromises in 2011, and could be unsuccessful attempts to compromise further CAs. Another incident happened at KPN/Getronics [128], which took over large parts of DigiNotar's business after its bankruptcy. An audit, which was conducted after the DigiNotar incident, revealed that several of the company's web servers were compromised four years earlier and the attack remained undetected although the attackers left attack tools for distributed denial of service attacks behind.

3.2.3 Wrap-up

The analyzed security incidents related to CAs have shown the practical relevance of the problems discussed in Section 3.1.3. The issuance of fraudulent certificates is evident, as is their use in attacks against Internet users. The multitude of CA incidents often in combination with poor security practices and inadequate reactions to incidents as well as the many cases of intentionally misbehaving CAs show that the system-centric trust model is inappropriate. Also, it was shown that revocation as implemented today cannot be considered sufficient in case of CA compromises and failures. In many cases, browser vendors reacted with emergency updates to blacklist fraudulent certificates. However, it was shown that this is not a general solution as it requires the complete knowledge of all fraudulent certificates. Furthermore, even emergency updates face delays until they can be published and are actually installed within browsers. Besides that, the too-big-to-fail problem was identified to prevent the revocation of compromised CAs' certificates, which would have lead to a much faster protection of the users than the mitigating measures taken to circumvent a complete revocation of the corresponding CA certificate. Although many incidents have been reported, one must assume that the number of incidents and intentional CA misbehavior is much higher [121]. This is due to poor reporting practices as observed in some of the described incidents. Furthermore, it must be assumed that governments that force CAs [63] to cooperate in the surveillance also prevent a public disclosure of such activities.

3.3 State of the art: Concepts for mitigation

In this section, we show that the problems described in Section 3.1.3 are unsolved so far. This is done by the examination of existing proposals that aim at the mitigation or the solution of these problems.

3.3.1 Proposals for the system-centric trust model problem

Many attempts exist to circumvent the problems related to the system-centric trust model. In the following we analyze the different proposals and show, that none of them completely resolves the problem in practice. The proposals have in common, that they limit the reliance on CAs by introducing additional mechanisms for certificate reconfirmations.

Public key pinning

Public key pinning [87, 170] means that a relying entity locally stores information that uniquely identifies a public key and relates it to a host name that has previously been accessed. Public key pinning exists in different forms regarding the stored information and how this information is bootstrapped.

Regarding the stored information it can be differentiated between certificate pinning, public key pinning and CA pinning. Certificate pinning means that either the certificate or the certificate's fingerprint is stored. Public key pinning refers to the storage of the public key or the key's fingerprint itself. CA pinning refers to the method, that instead of the end entity's key, the key (or certificate) of a CA is stored. This CA must be present in the certification path to the end entity's certificate.

Regarding the bootstrapping it can be differentiated between pre-installed information, bootstrapping by user interaction and the trust on first use (TOFU) approach. TOFU means that, when a URL is first accessed, the presented certificate (and key) is trusted and stored, while during subsequent connections, the same key is expected. Bootstrapping by user interaction means that the user is asked whether a certificate is trustworthy whenever a new certificate is received, while pre-installed information means that public keys are pre-installed, e.g., within software bundles.

The different approaches were invented to resolve problems of public key pinning, but none of them completely succeeded. One major issue is that public key pinning is a static approach, while key management on the web is highly dynamic. The assumption that underlies public key pinning is that a key assigned to a service remains constant, and a key change indicates fraud. However, there are many reasons why different keys occur reaching from certificate expiry over necessary key length increase to key exchanges because of a compromise. Furthermore, having a service with the same URL hosted on different servers like in content distribution networks (CDN) often results in different certificates that certify different keys and are issued by different CAs. In general, it is difficult to distinguish between legitimate and fraudulent key exchanges.

Also the bootstrapping and update mechanisms are problematic. Pre-installed keys are not a general solution for bootstrapping due to the lack of scalability. Thus, key pre-installation can only be applied on a very limited scope and is always application specific. An example for this is Google's Chrome browser [142], which is shipped with the public keys for Google services. This in fact led to the detection of several security incidents as was discussed in Section 3.2. For a general scope, user interaction or the TOFU approach is applied. Yet, in the first case this implies

warnings whenever a new key or a key exchange is observed. Studies [64, 30, 2] show, that warnings are nearly useless in practice and most users simply ignore them. This in turn makes pinning useless as it is likely that a fraudulent certificate will be accepted as a legitimate key change. Besides that, one cannot assume that users can distinguish between legitimate and fraudulent certificates.

The TOFU approach is the only viable approach when considering pinning for an unlimited scope of applications. As it implies that a possible attacker must be present during the first connection establishment to a website, it provides a clear security benefit by reducing the freedom of action for possible attackers. Yet, legitimate key changes are still problematic as these are not clearly distinguishable from fraudulent ones. Simply accepting a new key or falling back on the Web PKI would nullify the security benefits from pinning.

Finally, all pinning approaches must be able to recover from a successful attack or for example when a pinned key becomes unusable due to loss on the key owner's side. Then the question evolves how to inform the systems about such a case without enabling attackers to make use of such a mechanism.

There also exist several proposals [87, 100] which allow the operator of a web host to instruct clients which public keys they should accept in future connections. While still relying on the Web PKI for the initial connection establishment, the approaches handle legitimate key changes by informing clients in advance and installing backup keys for potential errors. This aims at allowing hard-fails, when a non authorized key change is detected. However, these proposals bear a high potential of server unavailability due to misconfiguration or mismanagement of the keys, which is left to server administrators. This concern is also reflected by the fact, that the proposals come with fall back mechanisms such as limiting the lifetime of pinning information in order to recover from configuration errors or even from pinning keys of an attacker during the initial connection establishment. Due to these drawbacks, such server assisted pinning solutions will rather stay niche solutions than being widely deployed.

Multi path probing and notaries

Multi path probing of certificates refers to the technique to contact a web server through different network paths to retrieve and compare the certificates served by the web server. The approach can detect man-in-the-middle attacks whenever there exists a path to the intended web server that is not controlled by the attacker because this leads to certificate mismatches. A clear advantage of multi path probing is the freshness of the used information, meaning that mismatches do not occur due to legitimate certificate changes. The difficulty is to establish such different paths as

the web infrastructure is not designed to allow that routing is controlled by clients. Thus, the connection to a server must be established starting from different entry points within the Internet.

Such different entry points are often realized by certificate notaries. Notarial solutions [145, 115, 171] may consist of single servers or a network of servers which can collaborate or operate independently. These notary servers can be queried to reconfirm certificates. The communication with the notary servers is secured by pre-installed keys distributed within the corresponding software bundles. When using notaries to evaluate the quality of certificates, trust is deferred from CAs to the notaries or rather a majority of notaries. As notary servers are distributed around the world, the resulting network paths to the target for which a certificate is to be reconfirmed differ from each other. This allows multi path probing of certificates. Additionally, notaries often maintain databases containing formerly observed certificates. This data is collected through passive monitoring of network traffic or active periodic monitoring of a given set of web servers.

On the one hand, multi path probing may suffer from false positives when different certificates for the same domain are served. This can happen when multiple servers are operated under the same domain as in CDNs. Then, a domain name does not always resolve to the same server, but the actual server is chosen according to load balancing rules or the geo location of the client. Well known examples for this are Google and Facebook. In such cases, the probing end point may differ during multi path probing and thus lead to certificate mismatches, even if all gathered certificates are legitimate. On the other hand, certificate databases cannot provide information about freshly deployed certificates or, when the reconfirmation of certificates of servers which have not previously been monitored are requested. However, the combination of certificate databases with multi path probing provides a robust set up, as the techniques compensate their mutual weaknesses.

While allowing the reconfirmation of certificates the main drawback of notaries is scalability and performance which prevents their broad application. For notaries, the same holds as for OCSP (cf. Section 3.3.2). Hard-fails must be enforced in order to provide protection against attackers that can potentially block connections to the notaries. However, this requires a highly available infrastructure, which the OCSP infrastructure fails to achieve since years. Furthermore, when multi path probing is applied, additional delays are introduced due to the connection establishment to a target server. These delays cannot be influenced by the notaries themselves. For example, measurements for the Crossbear notary [171] and Perspectives [115], which are used in the implementation of CA-TMS presented in Section 4.3, show varying round trip times between 0.2 and 1.2 seconds. A further drawback is the additional

network load, which is induced by multi path probing, which at least doubles the number of TLS connections to the target server.

We note, that multi path probing independent from notaries has also been proposed [3]. The proposal realizes multi path probing with the use of TOR. A second connection through the TOR network is opened to a server in order to retrieve the certificate and compare it to the certificate obtained in the normal connection. However, the approach can also not meet the performance requirements, as performance is one of the major problems of TOR. Another approach, to ubiquitously integrate multi path probing into the web infrastructure has been proposed in [B1] and is ongoing research of the author of this thesis.

Public logs

Public logs are publicly accessible servers that maintain databases of public keys and relate them to domain names. These servers can be used to look up the public keys of web servers, similar to a global phone book. Two experimental proposals currently exist, namely Certificate Transparency [98] and Sovereign Keys [123]. The main difference of these proposals lies in the scope of the public logs. Certificate Transparency aims at making all certificates issued by any CA of the Web PKI publicly visible to allow public monitoring of CAs. The goal is to enable domain owners to monitor the logs for fraudulent certificates issued for their domains in order to be able to initiate counteractive measures like revocation.

Sovereign Keys aims at the registration of additional, so called sovereign keys which are chosen and managed by domain owners themselves. Once a sovereign key is registered, it is used by the domain owner to cross-sign the server's TLS key or alternatively a CA key contained in the certification path to the server's key. Clients can then verify the authenticity of the server's key using the sovereign key registered for a domain.

Both proposals require a complex infrastructure for monitoring and auditing in order to prevent the manipulation of the public log servers and it is an open question who can operate this infrastructure in a reliable manner. Besides that, Certificate Transparency faces the same problem as OCSP and OCSP stapling. As long as it is not a common standard and implemented by all server operators and CAs, clients cannot reject connections simply due to the fact that a certificate is not found on a public log, otherwise many services become unavailable. However, this allows attackers to simply not report fraudulent certificates to prevent detection by public monitoring. The Sovereign Keys project introduces a very complex key management of the sovereign keys, which bears a high potential for misconfiguration

and the eventual unavailability of web servers.

While Google pushes forward the implementation of Certificate Transparency and implements a workaround by crawling the web to overcome the problem of limited support at least for extended validation (EV) certificates, the Sovereign Keys project is unmaintained since 2012. For none of the proposals a broad adoption is to be expected, which is at least for Certificate Transparency a basic requirement for its functioning.

DNS-based authentication of named entities (DANE)

An often discussed alternative to the Web PKI is the binding of certificates directly to DNS resource records. These resource records are to be secured against manipulation by DNSSEC. DANE [90] defines an additional TLSA-resource record. It allows zone administrators to specify the public keys of the web servers that are operated under the domain names managed in that respective zone. These keys are provided by the web server operators to the zone administrators. DANE allows to directly specify the web server's certificate (including self-signed certificates) or CA certificates which then need to be present within the certification path presented during the TLS handshake. Thus, DANE allows to alternatively or even exclusively validate server certificates based on the chain of trust given by the DNSSEC infrastructure. When comparing the DNSSEC infrastructure to the Web PKI, one can compare the zone operators to CAs. Other than realized in the Web PKI, DNSSEC has only one single trust anchor and each zone can only sign entries for a limited part of the domain name space, namely its direct child zones. This is seen as the biggest advantage over the Web PKI. However, despite being a strict hierarchy, DNSSEC has similar problems as the Web PKI. The trust management problem remains. On the one hand, each zone has limited power. On the other hand, there is absolutely no possibility to distrust one of the zones, as alternative trust paths are impossible. Thus, the system is even more susceptible to local law. Furthermore, it is important to know, that the zone operators are mostly the same players as the CAs, as e.g. Verisign for the top level domain .com.

Another drawback of DANE is the current deployment progress of DNSSEC, mainly on the client side [7]. Compared to the Web PKI, DNSSEC is a young technology and automatic support, e.g. to register server keys in resource records is missing. Furthermore, most DNS resolvers on client hardware like computers and smartphones are not capable to validate DNS records secured with DNSSEC [49]. Thus, the validation is in general delegated to some DNS name server, which simply indicates to the client if DNSSEC validation succeeded or not by setting a so

called DO-Bit (DNSSEC OK) in its answer. However, this requires that the name server must be trustworthy and the attacker must not be capable to manipulate the connection to the name server. This is in general not guaranteed.

From these findings we conclude, that DANE is a valuable addition for security and provides a means to reconfirm a web server's certificate obtained during the TLS handshake. However, it cannot be seen as a replacement for the Web PKI.

3.3.2 Proposals for the provision problem of revocation information

As shown in Section 3.1.3, revocation must be combined with hard-fails in order to be effective. Hard-fail means that a certificate must be evaluated as revoked if its actual status is unclear. Yet, even if revocation checking fails in only 1% of the connections, this prevents hard-fails from being implemented as the use of OCSP has shown over the past [158, 161].

OCSP stapling (cf. Section 2.2.3) theoretically resolves this and allows hard-fails [158], because the OCSP response is provided by the web server. This implies that revocation information for a service is available if the web server itself is. Yet, an attacker can remove the stapled OCSP responses. Once the deployment rate of OCSP stapling approaches 100%, the absence of stapled OCSP responses can be attributed to the presence of an attacker. However, according to the current SSL survey of Netcraft [162], OCSP stapling is only implemented on 24% of the web servers, which makes it impossible to decide whether an attacker blocked OCSP stapling or the web server simply does not implement it. Thus, up to now hard-fails are impossible.

The not yet standardized OCSP must staple extension [89] provides an opt in for hard-fails for server operators as it allows to anchor the OCSP stapling within the certificate. But still, protection relies on the deployment of OCSP stapling.

Because it is hardly assessable how long this will take, browser vendors like Google and Mozilla [141, 159] implement revocation pushing strategies called CRLSet and oneCRL. In both cases, these are aggregated CRLs which are pushed to browsers on a daily basis using the browsers' auto update features. Yet, as this approach does not scale, Google focuses only on high value domains, while Mozilla puts the focus on Sub CA certificates and extended validation (EV) certificates. While it is not reported, which web pages are actually covered, one can assume that these strategies are far from providing complete protection.

Besides that, blacklisting is another common approach to deal with the revocation problem. However, it requires all browser vendors to be informed of fraudulent

certificates which cannot be guaranteed as the incidents analyzed in Section 3.2 have shown. In Chapter 4 it is shown how user-centric CA trust management resolves the provision problem by enabling continuous monitoring of revocation information.

3.3.3 Proposals for the too-big-to-fail problem

Two closely related solutions exist for the too-big-to-fail problem. The first is to apply multiple signatures [52] by independent CAs. This would allow to revoke one of the involved CA keys, while certificates could still be verified based on the second signature.

The second solution is to add a trusted time-stamp to each certificate, generated by a TSA (cf. Section 2.2.7). The time-stamp would allow to securely identify which certificates have been issued before a revocation and which afterwards, thus certificates issued before a revocation of the CA certificate could further be considered valid.

Both approaches face several disadvantages. Firstly, they require the collaboration of independent CAs or of a CA with a TSA during certificate issuance which introduces overhead and requires the adaptation of currently deployed processes and business practices. Secondly, they introduce overheads into certificate validation. As both signatures on the certificate have to be verified, including revocation checking and path validation of independent certification paths, the overhead is doubled. Furthermore, this would require the adaptation of current standards, and their implementation on clients and web servers. These drawbacks have prevented the deployment of such solutions so far and the hurdles to realize such broad scale infrastructural changes will also do so in the future. Thus, these solutions are of limited practical relevance. We provide a solution for the too-big-to-fail problem in Chapter 6, which does not require infrastructural changes and is easily deployable within the current Web PKI.

3.3.4 Wrap-up

We have discussed the most prominent proposals for the mitigation or the solution of the problems described in Section 3.1.3 and it was shown, that none of these proposals comes without its own drawbacks, sometimes completely preventing a broad deployment. Other approaches are currently being deployed, however deployment is a tedious process in large scale systems like the Internet. This has also become evident in relation to the very serious Heartbleed vulnerability. It's removal simply required an update of the OpenSSL library on the affected servers. However, two

month after the bug was discovered, only half of the 600,000 affected servers had been patched. And security experts expect thousands of servers to be vulnerable even in ten years [169].

In Chapter 4 we present our solution for the trust management problem as well as the reliable provision of revocation information. The solution is based on the concept of user-centric CA trust management. It does not require a broad application in order to function. It makes use of several of the presented approaches without imposing perfect availability requirements in the integrated systems. It employs certificate pinning and notarial solutions, also additional mechanisms like Certificate Transparency or DNSSEC can be integrated as additional validation services used to reconfirm certificates.

3.4 The Web PKI from a user's perspective

In the following, it is shown that user-centric CA trust management is a promising solution for the trust management problem of the Web PKI. This is shown by analyzing real world browsing histories and examining how relying entities experience the Web PKI. It is shown that relying entities individually only require a small subset of the globally trusted CAs and that user-centric CA trust management has the potential to reduce the attack surface spanned by the entirety of globally trusted CAs by more than 95%.

3.4.1 The user-centric trust model for the Web PKI

In the user-centric trust model, trust decisions are made on a per user level. A central aspect of the user-centric trust model is that trust settings are individually set according to the requirements of the relying entity. Trust decisions further involve the relying entity's preferences and the subjective knowledge, the relying entity has collected during previous interactions.

For the Web PKI, a user-centric trust model implies user-centric CA trust management. Relying entities only trust the CAs they really need to validate the certificates they observe during the daily use of the Internet.

In the following, the potential of the approach is shown. Based on a user study, we evaluate how the currently deployed Web PKI is observed from a relying entity's point of view. We show, that the set of CAs relevant to a relying entity is indeed highly dependent on the individual browsing behavior. Our findings confirm that relying entities unnecessarily trust in a huge number of CAs, thus exposing themselves to unnecessary risks.

3.4.2 Web PKI user study - setup

For the pilot study whose results are presented in Section 3.4.3, we analyzed histories of 22 volunteers. To support the user study, the tool called Rootopia was developed. It automatically analyzes a relying entity's browser history and extracts the data regarding CAs the relying entity has observed in the past. Basically, it extracts the hosts that were accessed via TLS and extracts the related CAs. For more technical details on the tool please refer to [B4, 61].

An opt-in process was chosen for data collection, i.e., the users are required to actively hand over the results of the analysis of their browser histories. Besides that, we collected metadata using a questionnaire to be able to group the people into different categories.

Four persons provided two histories, either from different browsers they use in parallel, or different PCs. Thus, a total of 26 histories could be analyzed. All participants currently live in Germany, but have different cultural backgrounds. 16 of the participants originate from Germany, 2 from Poland, 2 from Morocco, 1 from Iran and 1 from China. The participants reach from IT experts to persons that only occasionally use a PC. The participants are between 25 and 57 years old. All of the participants either use Chrome or Firefox.

3.4.3 Findings

During the analysis of the collected data sets, user specific information as well as similarities and differences among user groups were derived. Table 3.1 shows aggregated numbers concerning history lengths and observed CAs. In the analysis we distinguish between true Root CAs and CAs that were seen both as Root and as Sub CAs (Root/Sub CAs). This resulted from cross-certification between Root CAs or the occasional inclusion of superordinate CAs into the certification path, even if one of the intermediate CAs is also present in the root store. As both Root and Root/Sub CAs must be present in the root store to be able to validate all observed certification paths, in the following we refer to them as the Root CAs.

Interestingly, none of the users – even those with a huge number of different TLS hosts – did see more than 22 different Root CAs, which is about 13.4% of the 164 CAs included in the Firefox root store. Furthermore, a maximum of 75 Sub CAs was reached. The absolute maximum of CAs in total seen by a single Internet user was 96, which is 6% of the 1,590 trusted CAs of the Web PKI. Even fewer CAs were found when only considering CAs that issued end entity (host) certificates. These CAs represent the minimum number of CAs that need to be trusted by a user to be

Criterion	Average	Min	Max
Duration of analyzed period (months):	18	4	38
Total number of TLS hosts:	168	12	636
Total number of TLS connections:	18,475	162	159,882
Total number of Root CAs:	10	4	14
Total number of Root/Sub CAs:	4	0	8
Root + Root/Sub CAs:	14	4	22
Total number of Sub CAs:	36	11	75
Number of CAs that issued end entity certificates:	33	8	68

Table 3.1: History sizes and numbers of observed CAs

able to verify all the certificates of the hosts he connected to. The maximum value of such CAs was 68 or 4.3% of the currently trusted CAs. The ratio of CAs issuing end entity certificates was in the span of 50%-75% of the total CAs found for the respective user and reached 63% on average.

Considering the total number of different Root and Sub CAs observed by the whole group of participants, namely the union of all sets of CAs, leads to 28 Root CAs and 145 Sub CAs. The numbers show that there is a high potential in limiting the number of trusted CAs. Furthermore, for certain user groups, there is a high overlap in the CAs (i.e. CAs that were observed by several persons). The overlap is significantly higher for Root CAs than for Sub CAs. This is reflected in the set union of Root CAs which is only 27% larger than the maximum number of Root CAs of a single user, while in the case of Sub CAs the set union consists of twice the number of Sub CAs seen by a single user. However, the significant differences in the numbers for different users – reflected in the minimum and maximum values – shows, that true minima for a single user can only be reached by individualization.

One influencing factor leading to a low number of different CAs is the fact, that there are only few large CA companies with a high market share in the certification business. However, the distribution we observed among those large players turned out not to be according to the market shares from the Netcraft SSL Survey [162]. Most significantly, VeriSign, Inc. is involved in more than 20% of the certification paths relevant for our user group, while it has only around 6% of the market share in the Netcraft Survey. In contrast, Go Daddy with more than 20% of market share was a Root CA in less then 4% of the certification paths in our data. This is another indication, that it highly depends on the individual browsing behavior of the users, which CAs are actually relevant for them.

The observed CAs were also grouped by country. It turned out that – compared to the total of 57 countries – CAs from only 14 different countries were relevant for the considered user set. The overwhelming majority of CAs is from the US (US) followed by Germany (DE), Great Britain (GB) and Belgium (BE). Considering the other countries, less than 5 CAs were observed from those and only by very few users.

Temporal evolution

In the following, we discuss our findings concerning the development of the individual views on the Web PKI over time according to the dates when related hosts were accessed. It turns out that the number of observed CAs does not grow linear but shows limited growth with high growth rates in the first few months. Considering Root CAs, the upper bound is reached after several months. However, growth rates depend on the intensity of Internet usage or rather on the number of TLS hosts a user connects to.

Considering users with high numbers of TLS hosts, the upper bound is reached faster than for users that only connect to TLS occasionally. For Sub CAs, the development is similar to the Root CAs, however, it is less significant. Thus, the number of Sub CAs tends to keep growing over a long time. To build user groups, we used the number of different TLS hosts averaged over the length of the analyzed time span. The average was approximately 9 hosts per analyzed month.

For the ten users that use TLS connections less intensively (i.e., who used less than 5 different TLS hosts per month), it takes a much longer time until the number of CAs approaches an upper bound. Yet, the upper bounds are strictly below the ones observed for users which use TLS a lot (i.e. the four users that used more than 18 hosts per month). On the other hand, there also exist users, that only connect to a very limited number of hosts but where the upper bounds on CAs are reached after very few months. This can be seen best in one data set, where the maximum of 4 Root CAs is reached after 3 months and is constant afterward (16 months). The picture for Sub CAs is nearly the same in that data set. A personal discussion showed, that the data belongs to a person using e-banking and e-commerce services, but besides that only occasionally surfs the Internet.

To summarize our findings on the development over time, we state that it is not possible to give a concrete number of months after which all relevant CAs have been seen and the number of CAs stagnates. This is highly depended on the individual browsing behavior. In many cases – due to the regular deletion of the histories – these are not long enough to derive the upper bound and the set of relevant CAs

for the respective user completely. Yet, in general, our observations show that the number of CAs tends towards an upper bound significantly below the total number of existing CAs. This in turn shows the potential for the possible attack surface reduction.

CA countries

As stated above, most of the observed CAs are from the US. The second most observed country in our set of participants is Germany. However, this is also a user group dependent outcome and results from the set of analyzed histories. A large part of the participants is either from the scientific community or students at a university. Building two groups, the first containing people with academic background and the second one without, shows that German CAs occur much less often in the second group. The percentage of German CAs is on average 18.3% of all observed CAs per user in the first, and only 7.1% in the second group. It results from the fact, that most universities have their own CAs, certified by the DFN Root CA. Those CAs are completely irrelevant for the non-academic users. The distribution of CAs over the other countries did not change significantly.

We also grouped the data into users that originate from Germany and those who do not. Yet, interestingly this did not have significant effects on the distribution over the countries. However, when considering single users, the relevant CA countries can depend on the country of origin as we observed it for a user from Poland (PL).

Considering all data sets, there are some country codes that were observed for most of the participants, yet where the respective CA was always one and the same. These are SE, ZA, NL, and IE.

For the remaining countries (KR, PL, UK, BM, FR, AU) no fix pattern is observable. From these, FR and BM are observed most often.

Relevance of CAs

The relevance of a CA for a user was measured based on the number of hosts related to the respective CA. Interestingly, the number of Sub CAs that are related to only one host lies between 20% and 60% of the total number of Sub CAs found for a user, and is about 43% on average. This shows that Internet users observe many CAs whose relevance is really low. Thus, it is highly questionable if the benefits for the user due to fully trusting in those CAs counterbalances the imposed risks, not speaking about the CAs a user never observes.

As it might occur that a single host is accessed extremely often by one user and thus the related CA becomes more relevant to him, we also measured the number

of visits, namely taking into account how often a host was accessed. As expected, the number of Sub CAs only observed during a single connection is lower. But still, rates of up to 38% of the total number of CAs for single users are reached and are 17.5% on average. This shows that many of the CAs are only observed by chance. Furthermore, our data shows that a user observes the CAs most relevant for him during the first months, while CAs which are found later are less relevant.

For each CA, we also averaged the CA's relevance over all users that observed the respective CA. It turns out, that there is a strong correlation between the number of users that observed a CA, and the averaged relevance of the respective CA. From these findings we conclude, that building user groups and taking the CAs which most users of that group have in common can be a good starting point to set up an individualized set of trusted CAs, e.g., for a user where no history data is available.

Number of CAs and overlaps

We computed the union of CAs for different user groups. To identify the similarity of the views on the Web PKI within a group, we computed overlaps in the CA sets, namely how many users have how many CAs in common. If not differently specified, in the following with overlap we mean the ratio of CAs that all group members have in common.

The group of the four users with most TLS hosts as specified in Section 3.4.3 jointly observed a total of 25 Root CAs and 108 Sub CAs. With 64% the overlap of Root CAs is twice the overlap of Sub CAs (31%). That shows, that the set of Root CAs relevant to a user is less dependent on the individual browsing behavior. This also holds for the other groupings we analyzed and is as expected, as the total number of existing Root CAs is nearly ten times smaller than the number of Sub CAs. Comparing the 25 Root CAs and 108 Sub CAs with the complete set of CAs jointly observed by all users, it turns out that the CAs seen by the users with most TLS connections make up for 89% of all Root CAs and 74% of the Sub CAs. Thus, most of the CAs required by the other users are also seen by the users with most TLS connections.

When comparing the groups of academic and non-academic users, the first observes significantly more CAs (27 vs. 19 Root CAs and 140 vs. 63 Sub CAs). This seems to result from the fact, that all the users with most TLS connections are also part of the academic group. The overlaps in the academic group are higher than in the non-academic group.

3.4.4 Discussion of the results

With the study we showed that the risk to be affected by CA malfunctions is unnecessarily high in the system-centric setting. It turned out, that the individual views on the Web PKI tend towards a stable individual set of CAs. The temporal evolution described in Section 3.4.3 actually shows different courses, thus confirming that the set depends on a user's individual browsing behavior. Our analysis indicates that a reduction of the number of trusted CAs by more than 95% is possible without restricting the respective user in his daily Internet use. We note, that a global limitation of the trusted CAs is no viable solution. The sets of required CAs are too distinct between different users. Thus, a global minimization of CAs cannot lead to an optimal solution. Furthermore, it would lead to interoperability problems and additional warnings whenever a certificate issued by an unknown CA is presented to the user.

We also found large differences in the relevance of the CAs, which leaves further room for improvement. However, it turned out that it is a challenging task to completely define the set of relevant CAs for an individual user. One problem is the unavailability of sufficient data about the user's browsing history, e.g. because of its periodical deletion. In such cases, grouping users and deriving group profiles can help to provide a starting point for the limitation. The study has shown that such groups exist, even though it is not possible to completely derive these groups from our data. This indeed would require a large scale study with users revealing a multitude of privacy sensitive information, which in turn prevents a broad participation in such studies. In Chapter 5 we provide a reputation system which exploits the fact that user groups exist, but realizes the grouping in a privacy sensitive manner without the need of predefined user groups.

Furthermore, mechanisms are needed to deal with CAs that are newly observed. Our data shows, that the number of CAs approaches a certain upper bound. However, new CAs can even occur after long time periods. Thus, views derived from past browsing behavior might always lack some CAs that are required in the future. In the next Chapter 4 we present our solution to locally and dynamically manage an individualized set of trusted CAs.

3.5 Conclusion

In this chapter, the defectiveness of the Web PKI was analyzed. The security and the attacker model have been presented. The Web PKI has been explained, along with its problems and how they support the described attackers. It was shown that

the problems are the system-centric trust model and the and shortcomings in the handling of revocation.

Afterward, CA security incidents have been analyzed. The described problems indeed make millions of Internet users susceptible to man-in-the-middle attacks. The large impact of security incidents can be explained with the system-centric trust model of the Web PKI, while the problems with revocation prevent fast and effective countermeasures. The CA security incidents reach from the erroneous issuance of certificates over CA system compromises to intentional CA misbehavior. Together with the observed reactions to these security incidents by the CAs themselves this shows the inadequacy of the system-centric trust model.

The scientific community is aware of the weaknesses of the Web PKI, thus many proposals for their mitigation have been made. These proposals have been analyzed and their strengths and weaknesses have been discussed. It was shown that the problems of the Web PKI are unsolved in practice. However, many of the proposed approaches can be combined as building blocks for user-centric CA trust management. Our solution will be described in the following chapter. The potential of user-centric CA trust management has been shown by a user study, where real world browsing histories have been analyzed. The study showed that the number of trusted CAs can individually be reduced by around 95%, thus reducing the attack surface of the Web PKI. Furthermore, as will be presented in the following chapter, the fact that relying entities individually only require trust in a small sub set of CAs of the Web PKI, user-centric CA trust management also enables the continuous monitoring of revocation information.

4 CA-TMS: User-centric CA trust management

In this chapter, we present our solution for the trust management problem and how to enable a reliable provision of revocation information. First, the realization of user-centric CA trust management complemented with continuous revocation monitoring is described. We call this system CA Trust Management System (CA-TMS). In the second part we show how CA-TMS is implemented. The last part of this chapter is concerned with the evaluation of CA-TMS.

In Sections 4.1 and 4.2 we present CA-TMS realizing user-centric CA trust management and continuous revocation monitoring. The components and mechanisms of CA-TMS and the parameters that control the system behavior are described. For these parameters, a parameter setting is presented.

The main idea of CA-TMS is to restrict the trust placed in CAs of the Web PKI to trusting in exactly those CAs actually required by a relying entity. To achieve this the certificate validation procedure is extended by trust validation. Trust validation is executed by a client program, the CA-TMS client. It communicates with the browser via a browser plugin. As input, trust validation gets a user dependent knowledge base called trust view. The trust view contains the user dependent information concerning required CAs and user preferences. The CA-TMS client also provides the algorithms for trust establishment, learning processes and information collection as well as bootstrapping. It allows the user to manage his trust view and to control whom he trusts and to which extent. Continuous revocation monitoring is implemented as an additional module. It becomes feasible due to the user-centric CA trust management and the related information collected in the trust view. The presented parameter setting is deduced from simulating CA-TMS based on real world browser histories and analyzing the system behavior for different settings.

Section 4.3 is concerned with the implementation of CA-TMS. We present the architecture and the modular design of CA-TMS. The implementation is available as open source software.

In Section 4.4 we evaluate CA-TMS regarding security and performance. It is shown that CA-TMS provides the aimed attack surface reduction of more than 95% compared to the standard system-centric setting as stated in Chapter 3. Also, it is shown that the solution is practical. We give a security analysis based on the attacker model presented in Chapter 3. To measure the attack surface reduction a metric is presented. This metric is evaluated on data obtained from the simulation of CA-TMS with real world browser histories using our proposed parameter setting. The performance of CA-TMS is evaluated in terms of the overhead induced by trust validation and additional certificate reconfirmations as well as continuous revocation monitoring. Section 4.5 concludes this chapter.

The contributions of this chapter were published as parts of [B2, B5]. This chapter extends the published contributions by continuous revocation monitoring and the implementation of CA-TMS. Furthermore, the evaluation of CA-TMS was revised and extended. The data sets used in the original evaluation were complemented with additional browsing histories collected after the publication of [B2].

4.1 Trust view and trust validation

User-centric CA trust management means that CA-TMS restricts the trust placed in CAs of the Web PKI to trusting in exactly those CAs actually required by a relying entity. The number of trusted CAs is individually reduced. To achieve this, the certificate validation procedure is extended by explicit trust validation in order to evaluate the trustworthiness of a connection according to the security model presented in Section 3.1.1. Different trust requirements for different applications are considered during trust validation. For example, there is a difference in the trust needed to visit a search engine and the trust needed to supply an online-shopping web site with credit card information. The core of CA-TMS is the trust view. It serves as a local and user dependent knowledge base for trust decisions. We illustrate the mechanisms for the establishment and the management of the trust view. Moreover, we implement learning processes and define decision rules for automated trust decisions. Trust is represented by employing computational trust models. The real trustworthiness of CAs is approximated by subjective probabilistic trust values.

CA-TMS is focused on applicability, thus we only use data which is initially available or is collected over time. However, the system is open for extensions with additional information sources. We build on existing techniques like public key pinning and certificate notaries and combine different mechanisms that complement

each other. Different from those existing mechanisms, CAs in the entity's trust view have different trust levels and may even be fully trusted depending on the context. Furthermore, trust evaluation is based on local experiences of the entity, not requiring recommended trust values embedded in certificates or the evaluation of certificate policies and expert opinions. Thus, our solution can work autonomously and does not require an additional check of every (new) certificate. CA-TMS provides a trade-off between overhead due to reconfirmations and solely relying on CAs. Furthermore, the management of local experiences guarantees that CAs are only trusted after they have previously been encountered and checked. A CA is only trusted when the entity needs this CA to authenticate a web service – independent from the CA's global reputation. This protects the entity from malfunctions of CAs that in general follow good security practices but are actually irrelevant for the entity itself.

From the analysis of the Web PKI and the existing mitigations for its weaknesses in Chapter 3, several constraints for the realization of user-centric CA trust management can be deduced. We present the challenges in the following.

4.1.1 Challenges

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

1. **Minimal user involvement**: an informed assessment of the quality of a CA's certification processes is beyond the capabilities of the average Internet user [30, 64]. Warnings should be omitted as far as possible, because users get used to and tend to ignore them, even leading to a weakening effect.

2. **Incomplete information on CA processes**: data on the quality of a CA's certification process might be incomprehensible and non standardized, incomplete, or not available at all [23]. Also, published policies are no guarantee for compliance [119].

3. **Incomplete information on the relying entity's requirements**: in general, the web services that an entity will contact in the future are unknown and thus also the CAs that are required to verify the certificates of these web services.

4. **Minimal latency for connection establishment and avoidance of blocking online verifications**: in order to be accepted and used by relying entities, the latency added to page loading must be kept as small as possible. Users in general do not tolerate waiting time [56] or blocked connections due to the unavailability of validation services (cf. Section 3.3.2).

4.1.2 Modeling trust validation

In the following we describe how trust validation is modeled. The final outcome of trust validation is an estimate for the *key legitimacy* of the public key pk certified in a certificate C. The key legitimacy of pk denotes the level of trust concerning its authenticity, i.e., whether pk in fact belongs to the identity contained in the subject field of C.

For a relying entity, in order to be convinced of the key legitimacy of pk, two things are required [36, 54, 75]. First, the relying entity must be convinced of the key legitimacy of the CA's public key with which the signature on C is verified. Second, the relying entity must trust the CA to issue trustworthy certificates. The latter is called *issuer trust* in the CA.

In this thesis the CertainTrust trust model is used to represent trust. Please refer to Chapter 2.4 for a detailed introduction and the definition of the related CertainLogic operators. CertainTrust together with CertainLogic provides the respective operators required in our context. Recall, that CertainTrust expresses trust-related information as opinions $o = (t, c, f)$, where t represents the trust, c denotes the certainty about the correctness of t and f defines an initial trust value which represents systemic trust.

With this, key legitimacy o_{kl} and issuer trust o_{it} are represented as independent opinions. The issuer trust assigned to a CA is further split into issuer trust for end entity certificates o_{it}^{ee} and issuer trust for CA certificates o_{it}^{ca}. The key legitimacy of a key is computed as the key legitimacy of the CA's key in conjunction with the issuer trust in the CA. In CertainLogic, the conjunction is realized with the AND operator.

Now let C be an end entity certificate binding the public key pk to the subject $\mathcal{E}$. C was issued by the CA $\mathtt{CA}$, i.e., it is signed with $\mathtt{CA}$'s private key and the signature can be verified with $\mathtt{CA}$'s public key $\mathsf{pk}_{\mathtt{CA}}$. Then, the key legitimacy of pk is denoted with

$$o_{kl,\mathsf{pk}} = o_{kl,\mathsf{pk}_{\mathtt{CA}}} \wedge o^{ee}_{it,\mathtt{CA}}.$$

The computation of the key legitimacy based on a certification path $p = (C_1, \ldots, C_n)$ of length $n > 1$, follows directly from chaining this rule, while for intermediate

certificates the respective issuer trust for CA certificates is used. For $1 \leq i \leq n-1$ let $o_{kl,i}$ be the key legitimacy of the public key in C_i and $o_{it,i}$ the issuer trust assigned to the subject in C_i (the subject in C_i is always the issuer of C_{i+1}). Then:

$$o_{kl,n} = o_{kl,1} \wedge o^{ca}_{it,1} \wedge o^{ca}_{it,2} \wedge \ldots \wedge o^{ee}_{it,n-1}.$$

The key legitimacy of keys distributed through an out of band channel can be assumed to be complete. Thus, the key legitimacy of the first key pk_1 in the path is $o_{kl,1} = (1,1,1)$ because it is the Root CA's key and distributed within the root store. As for the AND operator holds: if $o_A = (1,1,1)$ then $o_A \wedge o_B = o_B$, the formula for the key legitimacy can be simplified to:

$$o_{kl,n} = o^{ca}_{it,1} \wedge o^{ca}_{it,2} \wedge \ldots \wedge o^{ee}_{it,n-1}.$$

In the following we describe trust views as the user dependent knowledge base that contains the information trust evaluation is based on. Furthermore, we describe the algorithms for initialization, information collection, bootstrapping and trust validation in detail.

4.1.3 The trust view

For trust validation, entity $\mathcal{E}_1$ has a trust view View. The trust view is the local knowledge base of $\mathcal{E}_1$ and contains all previously collected information about other entities and their keys. It is built incrementally during its use for trust validation. View consists of:

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

The trusted certificates are all certificates that have previously been used to establish a trustworthy connection to another entity. The untrusted certificates are those certificates, for which the connection was evaluated untrustworthy. Furthermore, there is one public key trust assessments for each known pair $(\mathsf{pk}^*, \mathtt{CA}^*)$, i.e. for which a certificate binding pk^* to $\mathtt{CA}^*$ was contained in a previously evaluated certification path. A trust assessment represents all information collected for the respective pair during prior trust validations.

A public key trust assessment TA is a tuple $(\mathsf{pk}, \mathtt{CA}, S, o_{kl}, o_{it}^{ca}, o_{it}^{ee})$, where

- pk is a public key.
- $\mathtt{CA}$ is the name of a certification authority.
- S is a set of certificates. It contains all the certificates with subject $\mathtt{CA}$ and public key pk that have previously been verified by $\mathcal{E}_1$.
- o_{kl} is an opinion. It represents the opinion of $\mathcal{E}_1$ whether pk belongs to $\mathtt{CA}$ or not (key legitimacy of pk).
- o_{it}^{ca} is an opinion. It represents the trust of $\mathcal{E}_1$ in $\mathtt{CA}$ to issue trustworthy certificates for CAs (issuer trust in $\mathtt{CA}$ when issuing CA certificates that are verifiable with pk).
- o_{it}^{ee} is an opinion. It represents the trust of $\mathcal{E}_1$ in $\mathtt{CA}$ to issue trustworthy certificates for end entities (issuer trust in $\mathtt{CA}$ when issuing end entity certificates that are verifiable with pk).

In order to decide whether the connection to entity $\mathcal{E}_2$ is trustworthy, entity $\mathcal{E}_1$ runs the trust validation algorithm (cf. Section 4.1.5). First we describe how trust assessments are initialized.

4.1.4 Initialization of trust assessments

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

- $\mathsf{pk} = \mathsf{pk}^*$
- $\mathtt{CA} = \mathtt{CA}^*$
- $S = \{C\}$
- $o_{kl} = (1, 1, 1)$ if the CA is a Root CA, else $o_{kl} = \mathtt{unknown}$.
- The initialization of o_{it}^{x} for $x \in \{ca, ee\}$ is the following:
  1. If there exists $\tilde{\mathsf{TA}} \in \mathsf{View}$ such that $(\tilde{\mathtt{CA}} = \mathtt{CA}) \wedge ((\mathtt{CA}$ is a Root CA$) \vee$ (the issuer of C equals the issuer of one $\tilde{C} \in \tilde{S}$)), then set $o_{it}^{ca} = \tilde{o}_{it}^{ca}$ and $o_{it}^{ee} = \tilde{o}_{it}^{ee}$.

2. If there exists no such $\tilde{\mathsf{TA}}$ then:
   a) If for $1 \leq i \leq n$ there are trust assessments $\mathsf{TA}_i \in \mathsf{View}$ with $C_i \in S_i$, where the issuer of C_i is equal to the issuer of C, then compute $f^x = \frac{1}{n}\sum_{i=1}^{n} E(o^x_{it,i})$ and set $o^x_{it} = (0.5, 0, \min\{maxF, f^x\})$ for $x \in \{ca, ee\}$ and $maxF = 0.8$. $\min\{a, b\}$ denotes the minimum of the input values.
   b) Else set $o^{ca}_{it} = o^{ee}_{it} = (0.5, 0, 0.5)$.

The key legitimacy is set to complete ($o_{kl} = (1, 1, 1)$) for Root CA keys as these keys are confirmed via the root store. For other CA keys, key legitimacy is computed during trust validation as long as key legitimacy is `unknown`. During the evolution of the trust view, key legitimacy may be fixed and set to complete as soon as enough evidence has been collected. We discuss this in Section 4.1.6.

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

Step 2 provides an initialization mechanism if no prior information about the CA is available. If the new CA's key is certified by a CA that certified keys of several other CAs, i.e., there are siblings for which experiences have already been collected, we use the average over the expectations of the respective issuer trusts for initialization. The reason is that a CA evaluates a Sub CA before signing its key, and thus, these Sub CAs are assumed to achieve a similar level of issuer trust, like a stereotype [14]. While Burnett et al. apply machine learning techniques to identify the features that describe stereotypes, the solution presented here assumes that being certified by the same CA is the only relevant feature for stereotyping. Therefore, all known CAs that are certified by the same CA form a stereotype. We bound the initial trust value f by $maxF$ in order not to overestimate a CA's trustworthiness (cf. Section 4.1.8 for a discussion of the effects of the parameter choice for $maxF$). If also no siblings are available in the trust view, the issuer trust $o^x_{it} = (0.5, 0, 0.5)$ reflects that no experiences have been collected and that the CA may either be trustworthy or not.

Optimally, further information is collected for initialization. CA-TMS is open for such extensions. In Chapter 5, we describe how to realize a reputation system, which recommends the issuer trust of a CA to an entity based on the trust views of other entities. Further information could be gathered from policy evaluation as, e.g., proposed by Wazan et al. [69, 70]. A drawback of this approach is its need for some kind of expert or expert system to evaluate the certificate policies and

practice statements, because these documents cannot be processed automatically at the time being [23]. So far, no such services are available in practice. Yet, given such additional data, it can be mapped into an opinion and integrated into the initialization process.

4.1.5 Trust validation

Now the trust validation algorithm is described. It takes the trust view of entity $\mathcal{E}_1$ and a certification path for the certificate of entity $\mathcal{E}_2$ as input and computes the key legitimacy of $\mathcal{E}_2$'s public key to decide whether a connection established with $\mathcal{E}_2$'s key is to be considered trustworthy. The decision depends on the security criticality of the application that is to be executed between $\mathcal{E}_1$ and $\mathcal{E}_2$. The information available in the trust view may not be sufficient to complete the trust validation. In such a case, validation services are used as a fall back mechanism. Given a service provider as described in Chapter 5 is available, untrusted certificates can be reported to it. We include this optional step for completeness. For details on this functionality refer to Section 5.3.1. We present the detailed trust validation algorithm in the following:

Input:

- The certification path $p = (C_1, ..., C_n)$ without intermediary self-signed certificates
- The trust view View of $\mathcal{E}_1$
- A security level $l \in [0; 1]$ for C_n. l is selected by $\mathcal{E}_1$ and represents the security criticality of the application that is to be secured by the connection from $\mathcal{E}_1$ to $\mathcal{E}_2$. The higher l, the more security critical is the application.
- A list of validation services $VS = (\texttt{VS}_1, ..., \texttt{VS}_j)$ with outputs $R_i = \texttt{VS}_i(C) \in \{\texttt{trusted}, \texttt{untrusted}, \texttt{unknown}\}$ for $1 \leq i \leq j$ on input of a certificate C.
- *(optional)* A service provider SP (as described in Chapter 5) for the report of untrusted certificates.

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

The algorithm proceeds as follows:

1. If C_n is a trusted certificate in View then $R \leftarrow \texttt{trusted}$

2. If p contains a certificate that is an untrusted certificate in View then $R \leftarrow$ `untrusted`

3. If C_n is not a certificate in View then
   a) For $1 \leq i \leq n-1$ set pk_i to the public key in C_i and $\mathtt{CA}_i$ to the subject in C_i.
   b) Initialize the trust assessments for pairs $(\mathsf{pk}_i, \mathtt{CA}_i)$ for which there is no trust assessment in View (as described in Section 4.1.4). Store the new trust assessments in the temporary list TL.
   c) For $1 \leq i \leq n-2$ set $o_{kl,i}$ to the key legitimacy of pk_i and $o^{ca}_{it,i}$ to the issuer trust (for CA certificates) assigned to pk_i in View.
   d) Set $o_{kl,n-1}$ to the key legitimacy of pk_{n-1} and $o^{ee}_{it,n-1}$ to the issuer trust (for end entity certificates) assigned to pk_{n-1} in View.
   e) Set $h = \{\max(i) : o_{kl,i} = (1,1,1)\}$
   f) Compute $o_{kl,n} = (t,c,f) = o^{ca}_{it,h} \wedge o^{ca}_{it,h+1} \wedge \ldots \wedge o^{ca}_{it,n-2} \wedge o^{ee}_{it,n-1}$
   g) Compute the expectation $exp = E(o_{kl,n})$
   h) If $exp \geq l$ then $R \leftarrow$ `trusted`
   i) If $exp < l$ and $c = 1$ then $R \leftarrow$ `untrusted`
   j) If $exp < l$ and $c < 1$ then
      i. For $1 \leq i \leq j$ query validation service $\mathtt{VS}_i$ for C_n and set $R_i = \mathtt{VS}_i(C_n)$.
      ii. Set R_c to the consensus on $(R_1, \ldots, R_j)$, then $R \leftarrow R_c$.
   k) Update View (see Section 4.1.6 for details).
   l) *(optional)* If $R =$ `untrusted` trigger the report of p to SP

4. Return R

Security levels

Entity $\mathcal{E}_1$ assigns security levels to classes of applications according to their value-at-stake (cf. [69] for a similar approach). That means, $\mathcal{E}_1$ defines which security level the trust evaluation must achieve for the certification path in question in order to be accepted without further reconfirmation. Note that $\mathcal{E}_1$ does not rate the trustworthiness of applications.

A security level is a real number between 0 and 1. The higher the security level l is, the higher is the required key legitimacy for a connection to be evaluated trustworthy. The assignment of security levels is a subjective process and depends on the risk profile of $\mathcal{E}_1$, which is out of scope. We propose to apply three classes of security levels $l_{max} = 0.95$, $l_{med} = 0.8$ and $l_{min} = 0.6$ (cf. Section 4.1.8 for details on the choice of security levels and the associated effects). An exemplary assignment of applications to the security levels could then be l_{max} for online banking, l_{med} for e-government applications, and $l_{min} = 0.6$ for social networks.

Validation services

A certification path containing previously unknown CAs results in a low key legitimacy for the key certified in the end entity's certificate. On the one hand this is intended, as it leads to firstly distrusting in keys certified by unknown CAs. However, this is not necessarily due to malicious behavior, but due to a lack of information. Thus, whenever the key legitimacy is too low to consider a connection trustworthy, and the certainty is less than one, validation services like notary servers (cf. Section 3.3.1) are queried to reconfirm a certificate. Please refer to Section 4.3.1 for a list of currently supported certificate notaries. The communication with validation services is to be secured with keys distributed over out of band channels. This is achieved with distributing the public keys of the employed notaries within the CA-TMS software. Note that also solutions like certificate transparency or DNSSEC could be integrated as validation services. If a certificate is reconfirmed to be legitimate, the connection is considered trustworthy. If the validation services reply with `unknown`, i.e., it is unclear if the certificate is legitimate or not, the algorithm outputs `unknown`. Only in this case, the relying entity is asked for a decision.

In Section 3.3.1, latency introduced by validation services and their scalability were stated to be the main drawbacks of these solutions. As part of trust validation, these problems are circumvented. As will be shown in Section 4.4.3, reconfirmations are only required occasionally. Delayed page loading due to a necessary reconfirmation once every 10 days is acceptable taking into account the security gain by CA-TMS. Furthermore, the load on validation services is reduced drastically as only less than 0.7% of an average relying entity's TLS connections require a reconfirmation which mitigates the scalability problems. Additionally, only little information about the relying entity's browsing habits is leaked to validation services.

User interaction

As stated in Section 4.1.1, CA-TMS aims at minimal user involvement. While there exists a user interface for CA-TMS, where experienced users may change the system

parameters described in Section 4.1.8, and also have direct access to the trust view, this is not intended to be used during normal operation. Initialization of trust assessments, trust validation and the trust view update is performed autonomously in the background. It is built on passively collected local information (from past behavior and interactions) and input from validation services as well as the reputation system presented in Chapter 5.

User interaction during normal operation is limited to the specification of the security level the relying entity requires for the web site or web service he is about to open. While an automatized decision on the security level by the determination of the class of application together with a predefined rule set would be desirable, this is out of scope of this thesis. Possible solutions can, for example, be based on content filtering (as also used to detect phishing sites [74]) or based on analyzing the type of entered data (cf. [53]). For now, the required security level can be specified by the entity, using radio buttons that provide the different options for security levels from which an entity may choose as part of the browser's user interface.

Additional user interaction is only required as a fallback mechanism in cases where neither the local information is sufficient nor validation services can provide a decision on the acceptance of a certificate. Then, the relying entity must decide upon acceptance. In such cases no experiences are collected for the involved CAs, as the lack of expertise makes user decisions unreliable. The certificates in question are put on a watch list and experiences are collected after a later reconfirmation. This also allows the system to react to wrong decisions by the user and prevents collection of erroneous data. As the lack of expertise makes user involvement problematic, the `unknown` case needs to be avoided whenever possible by the use of an adequate set of validation services. The reputation system presented in Chapter 5 adds an additional information source to fasten bootstrapping. Thus, the amount of external reconfirmations and potential user interactions is reduced.

Apart from these cases no user interaction is required. In particular, relying entities do not actively provide their personal assumptions about the trustworthiness of certificates or CAs. This is also one of the main differences to other user-centric approaches, like PGP [75], where users have to state their opinion about the trustworthiness of other users and the legitimacy of their public keys or the Web of Trust [190], where users vote for the trustworthiness of provided content.

4.1.6 Trust view update

New information needs to be incorporated into the trust view to be available during future trust validations. Based on the output of the trust validation, either positive

or negative experiences are collected for the involved trust assessments. Repeated experience collection for the same activity must be prevented. Therefore, for each certificate it is checked whether it was contained in a certification path during earlier evaluations. Given a service provider as described in Chapter 5 is available, a recommendation for new CAs can be requested and incorporated into the trust view. We include this optional Step 3 for completeness. For details on this functionality refer to Section 5.2.1 in the following chapter. We present the detailed trust view update algorithm in the following:

Input:

- A certification path $p = (C_1, ..., C_n)$ without intermediary self-signed certificates
- A trust view View
- An output of the trust validation R
- A list of new trust assessments TL
- A boolean value $v \in \{\texttt{true}, \texttt{false}\}$ indicating whether C_n was validated by validation services or not
- A list of validation services $VS = (\mathtt{VS}_1, ..., \mathtt{VS}_j)$ with possible outputs $R_i = \mathtt{VS}_i(C) \in \{\texttt{trusted}, \texttt{untrusted}, \texttt{unknown}\}$ for $1 \leq i \leq j$ on input of a certificate C
- *(optional)* A reputation system $\mathtt{RS}$ (as described in Chapter 5) which outputs recommended issuer trusts $\mathtt{RS}(\mathsf{pk}, \mathtt{CA}) = (\tilde{o}^{ca}_{it}, \tilde{o}^{ee}_{it})$ on input of a pair $(\mathsf{pk}, \mathtt{CA})$.

Output: The updated trust view.

The algorithm proceeds as follows:

1. If $R = \texttt{unknown}$ then return View
2. For $1 \leq i \leq n-1$ set pk_i to the public key in C_i, set $\mathtt{CA}_i$ to the subject in C_i and set $\mathsf{TA}_i = (\mathsf{pk}_i, \mathtt{CA}_i, S_i, o_{kl,i}, o^{ca}_{it,i}, o^{ee}_{it,i})$ to the corresponding trust assessments.
3. *(optional)* If $(R = \texttt{trusted}) \wedge (v = \texttt{true})$ then $\forall \mathsf{TA}_i \in TL$ do:
   a) Request $\mathtt{RS}(\mathsf{pk}_i, \mathtt{CA}_i) = (\tilde{o}^{ca}_{it,i}, \tilde{o}^{ee}_{it,i})$ from $\mathtt{RS}$
   b) If $\mathtt{RS}$ did not return $\texttt{unknown}$ do:
   If $(i < n-1)$ set $o^{ca}_{it,i} = (0.5, 0, E(\tilde{o}^{ca}_{it,i}))$, else set $o^{ee}_{it,i} = (0.5, 0, E(\tilde{o}^{ee}_{it,i}))$.

4. If $R =$ `trusted` then
   a) For $1 \leq i \leq n-1$ do
      i. If $C_i \notin S_i$ then add C_i to S_i
      ii. If $(i = n-1)$ then update $o^{ee}_{it,i}$ with a positive experience, else if $(\mathsf{TA}_{i+1} \in TL) \vee (C_{i+1} \notin S_{i+1})$ then update $o^{ca}_{it,i}$ with a positive experience.
      iii. If $\mathsf{TA}_i \in TL$ then add TA_i to View.
   b) Add C_n to View as trusted certificate.
5. If $R =$ `untrusted` then
   a) Set $h = \{\max(i) : (\mathsf{TA}_i \notin TL)$ OR (the consensus of $(\mathtt{VS}_1(C_i), ..., \mathtt{VS}_j(C_i))$ $=$ `trusted`$)\}$.
   b) For $1 \leq i \leq h-1$ do
      i. If $C_i \notin S_i$ add C_i to S_i
      ii. If $(\mathsf{TA}_{i+1} \in TL) \vee (C_{i+1} \notin S_{i+1})$ then update $o^{ca}_{it,i}$ with a positive experience.
      iii. If $\mathsf{TA}_i \in TL$ then add TA_i to View
   c) If $C_h \notin S_h$ add C_h to S_h
   d) If $\mathsf{TA}_h \in TL$ then add TA_h to View
   e) If C_{h+1} is not an untrusted certificate in View then: if $(h < n-1)$ update $o^{ca}_{it,h}$ else update $o^{ee}_{it,h}$ with a negative experience.
   f) Add C_{h+1} to View as untrusted certificate.
6. Return View

Given a new certificate that was evaluated as trusted, a positive experience is collected for the issuer. In case the certificate was evaluated as untrusted, a negative experience is collected. For the calculation of the trust value alone, every negative experience cancels out a positive one and vice versa. In many situations, negative experiences should have a stronger influence on trust than positive ones: a certificate that failed the evaluation should not be trusted less but not at all. To meet these concerns, the affected untrusted certificate is immediately added to the list of untrusted certificates, in addition to the collection of a negative experience for the CA that issued the certificate. Thus, the certificate will never be accepted in

the future. Moreover, the actual impact of negative experiences on the final outcome of the trust evaluation can be controlled by using adequate security levels (cf. Section 4.1.5 and 4.1.8).

Example

An example of the evolution of a trust view is shown in Figure 4.1. It visualizes the experience collection process, starting with an empty trust view. The arrows represent observed certificates.

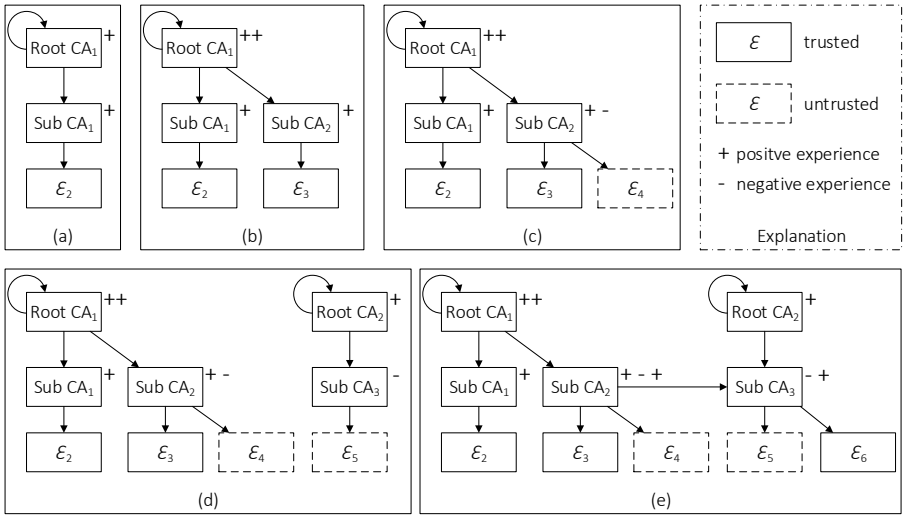


Figure 4.1: Evolution of the trust view

(a) CA-TMS obtains the path Root $CA_1 \rightarrow$ Sub $CA_1 \rightarrow \mathcal{E}_2$. Trust validation returns `trusted` for $\mathcal{E}_2$'s certificate. A positive experience is added to each involved CA.

(b) The path Root $CA_1 \rightarrow$ Sub $CA_2 \rightarrow \mathcal{E}_3$ is obtained. Trust validation returns `trusted` for $\mathcal{E}_3$'s certificate. A positive experience is added to each involved CA.

(c) The path Root $CA_1 \rightarrow$ Sub $CA_2 \rightarrow \mathcal{E}_4$ is obtained. Trust validation returns `untrusted` for $\mathcal{E}_4$'s certificate. A negative experience is added to Sub CA_2. However, the certificate Root $CA_1 \rightarrow$ Sub CA_2 was approved during prior observations, thus no negative experience is added to Root CA_1.

(d) The path Root $CA_2 \rightarrow$ Sub $CA_3 \rightarrow \mathcal{E}_5$ is obtained. Trust validation returns `untrusted` for $\mathcal{E}_5$'s certificate. Thus, the certificate Root $CA_2 \rightarrow$ Sub CA_3

must be checked. Assuming its reconfirmation, a negative experience is added to Sub CA_3, while a positive experience is added to Root CA_2.

(e) The path Root $CA_1 \rightarrow$ Sub $CA_2 \rightarrow$ Sub $CA_3 \rightarrow \mathcal{E}_6$ is obtained. Trust validation returns `trusted` for $\mathcal{E}_6$'s certificate. A positive experience is added to Sub CA_2 and Sub CA_3. Root $CA_1 \rightarrow$ Sub CA_2 was evaluated during prior observations, no new experience is added.

Fixing the key legitimacy

Different from the issuer trust, which might change over time, key legitimacy theoretically is constant once it is approved. From that point on, the issuer trust in superordinate CAs is of no further relevance. To consider this fact in the trust validation, key legitimacy is set to $o_{kl} = (1, 1, 1)$ as soon as enough evidence for the key legitimacy of a public key is available. We fix the key legitimacy after a CA's key was observed within several certification paths served by different web servers. This strategy is similar to multi path probing applied by certificate notaries. To realize the strategy, we introduce the parameter fix_{kl} and set $fix_{kl} = 3$, meaning that we fix the key legitimacy after collecting three positive experiences for the respective trust assessment. We refer the reader to Section 4.1.8 for an evaluation of the effects on the trust view for different parameter choices.

Cleaning the trust view

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

4.1.7 Bootstrapping

Despite the fact that an entity will often access the same services and see the same CAs repeatedly, it takes a certain time until enough experiences are collected such that the system may operate mainly autonomously (cf. Section 4.4.3 for details). Therefore, a bootstrapping procedure is required to face possible delays and usability problems due to the involvement of additional validation services as described in Section 4.1.5.

Bootstrapping is realized based on scanning the browsing history. This technique has already been used to evaluate the potential of user-centric CA trust management presented in Section 3.4.1. From the history, the hosts that have previously been accessed via a TLS connection can be identified and the respective certification paths can be downloaded. The paths are then used to bootstrap the trust view and collect information about the CAs relevant to the relying entity. Bootstrapping does not require additional algorithms. Trust evaluation along with certificate reconfirmations as described is executed on the certification paths obtained from history scanning one after another. This initial bootstrapping is only to be performed once and afterward, the system can mainly fall back on the collected experiences.

A limitation in the approach evolves from the fact, that many users delete their history – partly or completely – quite often for privacy reasons. In such cases, it is not possible to derive the CAs relevant to the user. Then, the trust view has to be bootstrapped in parallel to normal browsing with the drawback of high reconfirmation rates at the beginning of the trust view evolution. This problem is solved with the reputation system presented in Chapter 5.

4.1.8 Parameters and system behavior

The system behavior is controlled through different system parameters. In the following, the effects of these parameters on the system behavior is evaluated and a parameter setting is deduced. This is done based on real world browsing histories collected during the survey presented in Section 3.4.1.

Parameter n

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

Due to its impact on the certainty of an opinion, the parameter n of CertainTrust influences the development of the expectation of opinions during the course of collecting experiences. This is shown in Figure 4.2 for different values of n. While n has only little influence on the expectation during the collection of the first three to four experiences, n significantly influences the number of required positive experiences to reach expectation values approaching the upper bound of 1. That means, n in conjunction with the security levels can be used to adjust how fast CAs are considered fully trustworthy, while n concurrently has only minimal influence on the CAs that are approaching minimal security levels. Thus, increasing n mainly means

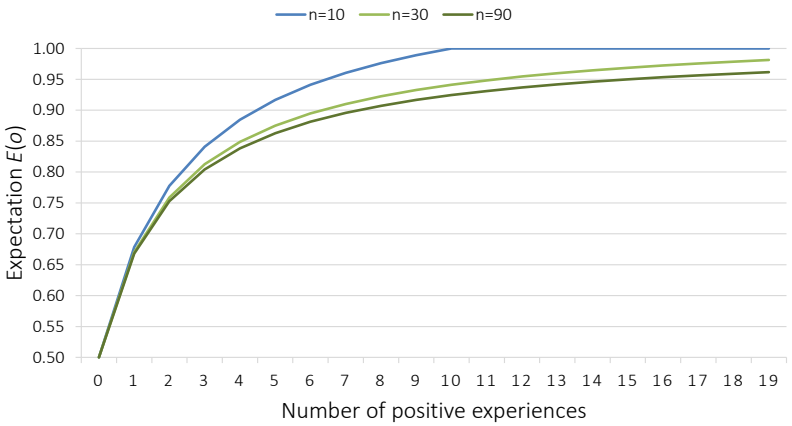


Figure 4.2: The influence of n on the development of the expectation of a CertainTrust opinion.

shifting CAs from the group of fully trustworthy CAs to medium trustworthy CAs while the group of CAs that reach the minimal security level remains unaffected.

Security levels

The expectation value of CertainTrust opinions is continuous. Thus, the required security level l for each application could also be chosen as any number between 0.5 and 1. We propose to assign applications to three distinct classes as already presented in Section 4.1.5. Doing so makes the system clearer and easier to use for potential users, as it reduces the complexity to decide which security level to require for an application.

The security levels to which a user may assign applications are $l_{max} = 0.95$, $l_{med} = 0.8$, and $l_{min} = 0.6$. Once the the expectation of the derived key legitimacies for certificates issued by a CA exceeds l_{max}, the CA is referred to as fully trustworthy (medium or minimally trustworthy respectively).

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

Parameter fix_{kl}

In Section 4.1.6, we proposed to fix the key legitimacy of a CA's key after observing a CA's key within fix_{kl} certification paths. After the fixation of the key legitimacy, superordinate CAs have no further influence on the evaluation of certificates issued by the CA. Then it relies solely on the experiences made with the CA directly. We propose to fix the key legitimacy after three positively evaluated encounters, i.e., set $fix_{kl} = 3$.

Similar to n, increasing fix_{kl} mainly reduces the number of fully trustworthy CAs, while the number of medium and minimally trustworthy CAs grows. This is shown in Figure 4.3. It depicts, exemplary for the most evolved trust view from the data set of the survey presented in Section 3.4.1, how the distribution of hosts associated with CAs reaching the different trustworthiness levels changes for values from $fix_{kl} = 1$ up to $fix_{kl} = 80$. The distribution is measured for the bootstrapped trust view after observing 604 different TLS enabled hosts.

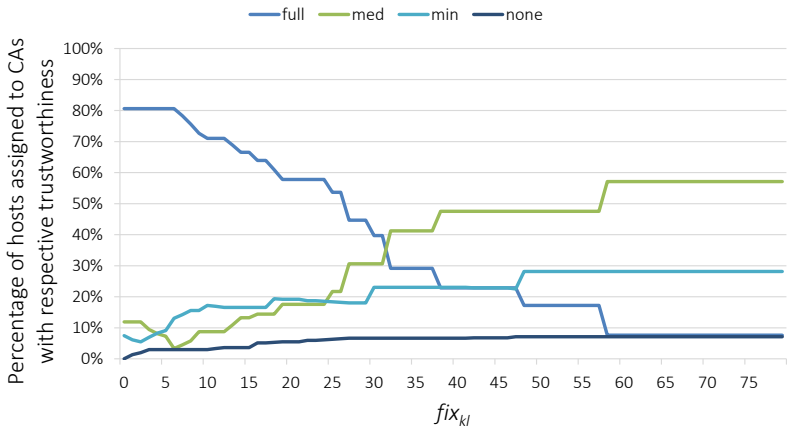


Figure 4.3: Percentage of total hosts that are assigned to fully, medium, mimimally trustworthy and untrustworthy CAs for different values of fix_{kl}. For one exemplary trust view.

The effects yielded by fixing the key legitimacy is explained by the fact that the expectation for the key legitimacy of issued certificates is not lowered by the trustworthiness of superordinate CAs (which is due to transitively lowering the key legitimacies along the certification path).

The value of fix_{kl} also has influence on how fast a CA achieves its previous trustworthiness level after its key was changed. Directly after the key renewal, the derived key legitimacy of certified end entity keys will be low, as the key legitimacy of the CA's key is computed based on the path from the root to the CA's (new) certificate. After fix_{kl} encounters of the CA's key, the previous trustworthiness level is

reestablished, i.e. the system solely relies on the experiences collected for the CA in question.

Stereotyping and parameter *maxF*

Stereotyping is a means to derive trust for newly observed CAs based on experiences collected for other CAs. This, on the one hand, allows to trust in CAs never observed before. Thus, the number of required reconfirmations is reduced. On the other hand, the anticipated trustworthiness of the CA is not based on the CA's behavior directly and information gained from stereotyping should not be overestimated to prevent threats. Threats may evolve when a less trustworthy CA profits from the reputation of its siblings. Therefore, we limit the influence of stereotyping by selecting the parameter $maxF = 0.8$. This limits the certificates that are directly trusted due to stereotyping to low security applications. The difference in the functioning of stereotyping compared to fixing the key legitimacy is that the fist affects newly observed CAs and the certificates issued by those, while the latter affects newly observed end entity certificates issued by already known CAs.

4.2 Continuous revocation monitoring

In Section 3.1.3 the lack of a reliable provision mechanism for revocation information in the face of an attack was identified for the Web PKI. Together with the currently implemented soft-fail methodology in browsers, this may render revocation completely useless. In Section 3.3.2, OCSP stapling was discussed as a potential solution to this problem. Yet, it was shown that because of deployment issues and the related impossibility to enforce OCSP stapling by client software, this does not solve the problem.

We now present a mitigation to this problem that can be implemented without external support and without the requirement of a broad deployment and infrastructural changes.

As described in Section 4.1 all certificates trusted by a relying entity are collected within his trust view. These certificates contain the CRL distribution points or OCSP server addresses. Thus, the data collected within the trust view enables to continuously monitor the revocation status for known hosts as well as known CAs. Thus, the revocation status of certificates is already available at the time of connection establishment in most cases.

4.2.1 Functionality

To realize continuous revocation monitoring, CA-TMS maintains a list of revocation access points. Whenever a new certificate is added to the trust view as trusted certificate, the revocation access points are extracted from the certificate's extensions (`CRL Distribution Point` and `Authority Information Access` extensions). These access points are stored in the list. Concordant access points are aggregated, which mostly happens for certificates issued by the same CA. Once a certificate expires or is considered untrusted, the revocation access points of the certificate are discarded from the list. This procedure keeps the list up-to-date.

On a daily basis, CA-TMS iterates through the list and fetches revocation information for the certificates in the trust view. Given one of the certificates is revoked, the status of the certificate is set to untrusted within the trust view. If an OCSP server or a CA's server that provides the CRLs is unavailable at the time of revocation checking, the request is delayed and later retried.

4.2.2 Advantages of revocation monitoring

The main advantage of this continuous revocation monitoring is, that it decouples fetching of revocation information from browsing. It runs in the background, thus does not block page loading, which makes longer round trip times acceptable. Whenever a relying entity accesses a web server, revocation information is already available. Furthermore, despite not being impossible, it is much harder for an attacker to block revocation checking as there is no direct relation between service access and the retrieval of revocation information. Other than revocation pushing strategies as discussed in Section 3.3.2, revocation monitoring covers all certificates relevant to the relying entity, including CA certificates. Note that downloading CRLs from all trusted CAs contained in the trust view additionally covers certificates issued by these CAs but not yet observed by the relying entity.

4.3 Implementation of CA-TMS

This section is concerned with the implementation of CA-TMS. We present the architecture and the modular design of CA-TMS. The source code is available under the Apache Software License 2.0 [181] at `https://github.com/ca-tms/CA-TMS`.

Trust validation is executed by a client program, the CA-TMS client. It communicates with the browser via a browser plugin. The CA-TMS client is implemented in Java [168]. As browser, the Firefox browser was chosen and an according plugin

was implemented that realizes the communication between browser and CA-TMS client. The CA-TMS client is designed to be application independent. The setup is displayed in Figure 4.4.

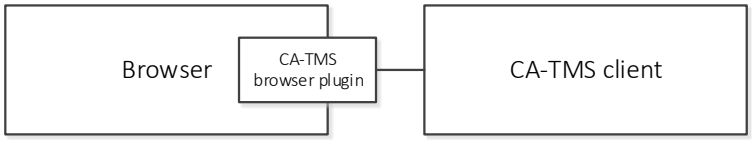


Figure 4.4: CA-TMS setup

The CA-TMS client maintains the user's trust view and implements the algorithms for trust validation, learning processes and information collection as well as bootstrapping and revocation monitoring. Additionally, it allows the user to manage his trust view and to configure the user preferences through a user interface.

Here, we focus on the system design and give an overview on the implemented functionality. For implementation details and a manual for installation we refer the reader to the project web page.

4.3.1 The CA-TMS client

In the following the architecture of the CA-TMS client is described. The client is organized into different layers that group related functionality. The layers themselves are subdivided into different components. We first describe the top level architecture and summarize the functionality of the layers. Afterwards, we give a detailed specification of each layer and the contained components. The design is based on the *Layered Application Guidelines* of the *Microsoft Application Architecture Guide* [154].

The high level architecture is shown in Figure 4.5. It is structured into five layers and a cross-cutting (CC) module. The layers are the presentation layer (PL), the services layer (SL), the business layer (BL), data access layer (DAL) and the support services access layer (SSAL). The PL contains all components to provide user interaction. It enables user input and user information. The SL contains all components to expose the functionality of the CA-TMS client to other applications. The BL encapsulates the business logic of the CA-TMS client and implements its core functionality. The DAL comprises the components related to data access and representation of user-specific data. The SSAL provides access to support services and implements components that enable the consumption and processing of data provided by support services. The CC implements functionality that spans layers.

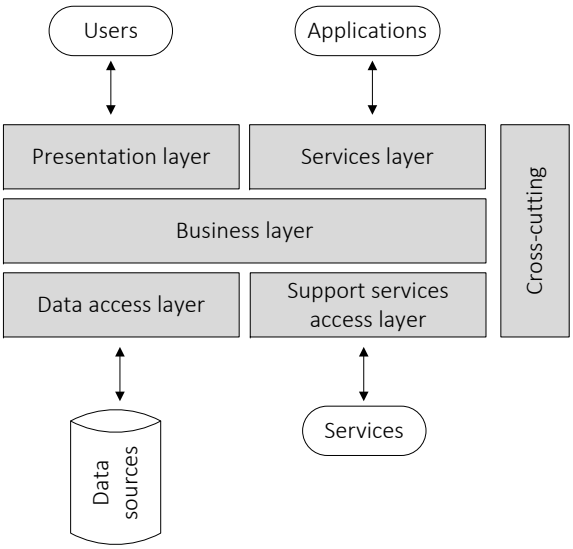


Figure 4.5: High level architecture of the CA-TMS client

In the following we describe the layers and explain which functionality they implement.

Presentation layer

The PL provides the means for user interaction, like input fields and dialogs. The PL is divided into a graphical user interface (GUI) component and a presentation logic component. The GUI implements the visual elements of the application like buttons and message dialogs which are used to display information to the user and accept user inputs. The GUI of the CA-TMS client is a management GUI, which is not required during normal use.

The different processes for user interaction are encapsulated within the presentation logic component. Thus, the presentation logic component defines the logical behavior of the client during user interaction. Furthermore, it defines how data from the underlying layers is presented to the user. This is realized in a platform independent way using the interfaces provided by the GUI component. Furthermore, it manages how the application reacts to user inputs.

The PL realizes a management GUI for the CA-TMS client and allows the user to configure the CA-TMS client and execute initialization processes like bootstrapping. It also provides the possibility to export the user's trust view or to import an existing trust view into the system. User interaction directly related to browsing, such as warning dialogs and reconfirmation requests are realized through the browser plugin, which extends the browser's user interface.

Services layer

The SL contains all components to expose the functionality of the CA-TMS client to other applications. It defines how the client and other applications interact. The service interfaces and the message types are defined here. On the other hand a common interface is provided to the BL. Thus, the SL abstracts from the actually supported service consumers.

The CA-TMS client exposes its services to other application via a web server binding. The CA-TMS client implements a web server that can be queried by the browser plugin with JSON encoded messages over HTTP.

Business layer

The BL encapsulates the business logic of the CA-TMS client. It implements the core functionality of the client such as trust evaluation for given certification paths and the management of the trust view. In particular, it implements the trust validation and trust view update algorithms. Furthermore, the logic for continuous revocation monitoring is implemented in this layer. The BL exposes the functionality of the client to higher layers as PL and SL. For data access or information retrieval from external services, the BL falls back on functionality provided by the lower layers, namely the DAL and the SSAL. Furthermore, it manages escalation rules if no decision can be made based on the available data. For example, the user can be triggered and the decision can be requested if no automatized decision is possible. For trust representation and computation the CertainTrust library [180] is integrated into the client. It provides the necessary operators for combination and aggregation of CertainTrust opinions.

Data access layer

The DAL comprises the components related to data access and representation. Besides that it handles import and export of trust views for backup and recovery. The DAL provides a common interface to retrieve and store user data from and to different sources.

For example, through the DAL, the components of the BL can access data on internal or external data sources. The CA-TMS client stores the trust view and the user's preferences within an SQLite [177] database. The encapsulation of data access within the DAL allows a convenient integration of other storage technologies, as the actual data source is transparent for the other client components. An example could be a remote data base server or a cloud storage.

Support service access layer

The SSAL provides common interfaces to access external support services such as certificate notaries for the reconfirmation of a certificate or OCSP servers for revocation checking. The SSAL allows to access different services in a standardized manner. It manages the semantics of communication with the external services. The SSAL is used by the BL components. Access to notary services is encapsulated in a separate library, which is developed in a sub project and available at `https://github.com/ca-tms/sslcheck`. At the time of writing, it implements connectors for the Crossbear notary [171], Perspectives [115], Convergence [145], ICSI [133] and SignatureCheck [175].

The encapsulation of external service access within the SSAL allows to extend CA-TMS with additional services in a convenient way. In particular, the integration of CA-TMS service providers as described in Chapter 5 can be realized within the SSAL.

Cross-cutting

The client components use cryptographic functions and communication protocols. This functionality is encapsulated within the respective cross-cutting components.

4.3.2 The browser plugin

The browser plugin has been implemented as a Firefox extension. The extension manages the communication between the browser and the CA-TMS client. It implements an SSL Listener, that is triggered whenever a web site is opened via the HTTPS protocol. Path validation is left to the browser implementation. If path validation has succeeded, the extension extracts the certification path and passes it to the CA-TMS client for trust validation. The loading of the web page is blocked until the validation result is obtained from the CA-TMS client. If trust validation succeeds, the web page is loaded. Otherwise, a warning message is displayed which informs the user about the outcome of trust validation and allows to temporarily override the trust validation result if required.

Additionally, the extension provides a simple user interface to set the required security level during browsing. The security level is transferred to the CA-TMS client when trust validation is requested.

4.4 Evaluation

In this section, we evaluate CA-TMS regarding security and performance. We start with the analysis of direct attacks against CA-TMS in Section 4.4.1. Afterward in Section 4.4.2 we present an analysis of the reduction of the attack surface. Then in Section 4.4.3, the performance analysis of CA-TMS is presented.

We show, that CA-TMS is robust against the manipulation of an entity's trust view. Furthermore, a metric is presented to measure the reduction of the attack surface. With this metric it is shown that CA-TMS provides an average attack surface reduction of more than 95% compared to the system-centric setting. The performance of CA-TMS is evaluated in terms of the overhead induced by trust validation and additional certificate reconfirmations as well as continuous revocation monitoring. It is shown that the use of CA-TMS does not interfere browsing in practice.

The attack surface and performance evaluation are based on simulations using real browsing histories. The data was collected using the tool Rootopia which was developed for the user study presented in Section 3.4.2. From the analyzed histories, the web services (hosts) accessed via TLS connections have been extracted along with the respective certification paths. These paths are available sequentially ordered based on the date when they where first accessed. This sequentially ordered paths allow to simulate the evolution of the respective trust views. Note that in the simulation, the collection of experiences is limited to positive experiences. Due to only collecting positive experiences, the resulting opinions on the issuer trusts form an upper bound for the actually derived trustworthiness in real life applications.

The original evaluation of CA-TMS published in [B2] was based on twenty data sets collected during the original user study. After the publication, additional data was collected during a second user study. In total, data about browsing histories from 64 different entities were collected. The evaluation was revised and extended to the complete data set. We note, that the key findings from [B2] remain intact.

4.4.1 Attacks against CA-TMS

The security analysis is based on the attacker model presented in Section 3.1.2. Recall that attacker $\mathcal{A}$ is an active man-in-the-middle attacker on the connection between relying entity $\mathcal{E}_1$ and web server $\mathcal{E}_2$. $\mathcal{A}$ can generate certificates that are signed by a CA of the Web PKI. The systems of $\mathcal{E}_1$ and $\mathcal{E}_2$ are assumed not to be compromised, nor can $\mathcal{A}$ break the employed cryptographic algorithms. Also, the validation services are assumed to function correctly.

This section is focused on attacks against specific components of CA-TMS. The attacks aim at the manipulation of $\mathcal{E}_1$'s trust view. Following from the attacker model, the intention behind these attacks is to prevent detection when $\mathcal{A}$ attacks $\mathcal{E}_1$'s communication employing a fraudulent certificate. Following the analysis framework of Hoffman et. al [31], the attack vectors are identified based on the separation into formulation, calculation, and dissemination components of CA-TMS.

Formulation resembles the reputation metric and sources of input. A CA's reputation is stored as opinion, which is updated by positive or negative experiences. The information source are observed certification paths along with the responses of validation services. Thus, $\mathcal{A}$ can positively or negatively influence the trust in a CA if he succeeds in serving manipulated certification paths.

Calculation concerns the algorithms that derive trust from the input information. In CA-TMS, these algorithms are deterministic and executed locally. Thus, as $\mathcal{E}_1$'s system is assumed not to be compromised, $\mathcal{A}$ cannot influence the calculation other than by injecting manipulated input data.

Finally, *dissemination* concerns all transfer of data between system components. The communication with validation services is secured using pre-established keys, and cannot be manipulated. $\mathcal{A}$ may block or disturb the communication, which is only relevant in case CAs are observed anew, or not enough information has previously been collected. However, the unavailability of validation services does not allow the manipulation of the trust view, even in case $\mathcal{E}_1$ manually accepts a manipulated certification path (cf. Section 4.1.5).

Attack vectors

Following from the analysis, the possibilities to manipulate and influence the trust evaluation of CA-TMS under the given assumptions is limited to injecting false input data. We now explain the attack vectors $\mathcal{A}$ might use to inject false data into $\mathcal{E}_1$'s trust view, and explain the respective protection mechanisms. We also discuss social engineering which may serve as a auxiliary attack vector to support other attacks.

Certification path manipulation To directly inject information into the trust view, $\mathcal{A}$ needs to inject manipulated certification paths into $\mathcal{E}_1$'s communication during the connection establishment. In order to inject positive experiences (for a CA of which $\mathcal{A}$ controls the private key), he must inject the CA's certificate into certification paths such that trust validation succeeds.

To inject the CA's certificate, $\mathcal{A}$ can either replace the whole path, by generating a new certificate for the web service with the CA's key, or by issuing a certificate for

one of the intermediary CAs contained in the original path sent by the web server. Therefore, this certificate replaces only the part from the Root CA to the newly issued Sub CA's certificate. In both cases, standard path validation succeeds.

The first option is exactly from what CA-TMS shall protect and will be analyzed in detail in Section 4.4.2. $\mathcal{A}$ may only succeed if the compromised CA is already trusted by $\mathcal{E}_1$. This means that this attack vector may only be used to increase the trust in CAs that are already trusted. The second path manipulation option will not be detected as the end entity certificate remains unchanged. Yet, this way the attacker can only increase the corresponding issuer trust for issuing CA certificates.

Injecting fraudulent certificates in order to generate negative experiences is impossible for CAs $\mathcal{A}$ does not control, as he cannot generate certificates in the name of CAs of which he cannot access the keys.

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

The direct influence of social engineering attacks on the reputation of a CA is limited, as long as a relying entity does not use the management GUI of the CA-TMS client to manipulate trust scores. Such an attacker cannot be prevented by technical means and is therefore not considered further. However, we note that to prevent such attackers educating relying entities to be security aware is indispensable.

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

Another possibility for an attacker is to influence a relying entity to setting the required security level l to a low value, which would increase the probability for the attacker's certificate to be accepted. Therefore, CA-TMS recommends a minimal security level such that CAs are never trusted without at least one legitimate observation. To prevent misconfiguration, automatic detection of the required security level according to a specific rule set in conjunction with the content of a web page is an interesting future research direction.

Also, an attacker could try to gain information about the relying entity's browsing behavior, for example by interviewing the relying entity. This in fact would help the attacker to (partially) derive the relying entity's trust view and subsequently distinguish between those entities that are attackable with certificates issued by a certain CA and those that are not. Anyway, these sorts of social engineering attacks are limited to single relying entities and are thus costly to execute.

Attacker goals and defenses

The previous subsection discussed the attack vectors that $\mathcal{A}$ can use to manipulate trust views. In this section, specific attacker goals, their possible realization, and how they apply to CA-TMS are discussed.

Self-promoting Self-promoting describes actions of $\mathcal{A}$ towards making him or a CA under his control appear more trustworthy. This in fact is the attacker goal with highest relevance as it increases the success probability of $\mathcal{A}$ when finally employing fraudulent certificates to attack secure communication.

A self-promoting attacker can approach his goal by injecting manipulated certification paths. As shown above, certification path manipulation only works for the issuer trust concerning the issuance of CA certificates. This might subsequently allow $\mathcal{A}$ to issue CA certificates with high key legitimacy. However, he cannot influence the issuer trust concerning the issuance of end entity certificates, which in fact is required to benefit from the attack. This is only possible for CAs that are already trusted making the attack needless.

Slandering In opposition to a self-promoting attack, slandering aims at lowering the reputation of a specific CA. As $\mathcal{A}$ does not directly benefit from decreased trust in CAs, he might only aim at disturbing the proper functioning of CA-TMS. As shown above, $\mathcal{A}$ cannot inject fraudulent certificates on behalf of CAs he does not control, thus he cannot utilize the attack vector of manipulated certification paths to inject negative experiences.

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

If $\mathcal{A}$ controls a CA with negative reputation, he might want to give this CA a new identity to issue certificates that will appear trustworthy. However, whitewashing does not apply to CA-TMS. Newly observed certificates are not automatically

trusted and thus, $\mathcal{A}$ 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.

4.4.2 Attack surface evaluation

In this section, we evaluate how CA-TMS reduces the attack surface of the Web PKI. When $\mathcal{E}_1$ uses CA-TMS, trust validation must succeed for a presented certification path. Otherwise, validation services are queried and a potential attack is detected. Recall that $\mathcal{A}$ might block the access to validation services to prevent a definite detection. However, in such cases the connection is temporarily blocked and $\mathcal{E}_1$ is informed about the suspicious certificate.

Trust validation can only succeed if the certificate is already a trusted certificate in $\mathcal{E}_1$' trust view View or if the involved issuers are contained in View. Furthermore, those CAs must be considered sufficiently trustworthy for the required security level of the application. Thus, $\mathcal{A}$ must compromise a CA with sufficiently high issuer trust in order to be successful.

Attacking a specific group of entities requires the compromise of a CA with a sufficiently high issuer trust in each of the group member's trust views. The same holds when attacking a specific service. In this case, the relevant set of trust views are those of the group of service users. Otherwise, $\mathcal{A}$ risks an immediate detection of the compromise when the validation services are triggered. As shown in Section 3.2, several past CA compromises have been detected in exactly that way where the compromised CA was not trusted by the Chrome browser. As trust views are specific to individual entities and not publicly visible, it is hard to identify such a sufficiently trusted CA. Even if the identification is possible, it is questionable if $\mathcal{A}$ can purposefully compromise that CA. In Chapter 5, a push service for CA warnings is presented to also protect entities whose trust view already contains the CA controlled by $\mathcal{A}$ as a trusted CA. The detection mechanisms is shown to be effective even if only a small fraction of the targeted entities does not trust the compromised CA.

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

Measurement of the attack surface

The attack surface is individually reduced by CA-TMS by limiting the number of trusted CAs. The attack surface is a measure for the individual risk to rely on a fraudulent certificate. We measure the effectiveness of CA-TMS by adapting the metric of Kasten et al. [41], which measures the attack surface as $AS = \sum_{\mathtt{CA} \in \mathsf{PKI}} dom\,[\mathtt{CA}]$, where $dom\,[\mathtt{CA}]$ is the number of domains which $\mathtt{CA}$ is allowed to sign. PKI describes the set of all CAs which are part of the Web PKI. We adapt the metric to:

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

with

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

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

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

where dom_{max}, dom_{med}, and dom_{min} are the respective numbers of domains for which the relying entity $\mathcal{E}_1$ requires a maximal, medium or minimal security level. The input View represents the trust view of $\mathcal{E}_1$. A CA is contained in View, if it contains the according trust assessment. Note that CAs not contained in View are not considered, as these are not trusted by $\mathcal{E}_1$ to sign any certificate. $o_{kl,ee} = o_{kl,\mathtt{CA}} \wedge o^{ee}_{it,\mathtt{CA}}$ is the derived key legitimacy of keys certified by a certificate that was issued by $\mathtt{CA}$. The key legitimacy $o_{kl,\mathtt{CA}}$ of the CA's key depends on the certification path as described in Section 4.1.5.

Further it holds: $dom_{max} + dom_{med} + dom_{min} = dom$. dom describes the total number of validly signed domains, i.e., for which a valid TLS certificate exists. Note that in our metric dom is not parametrized by the respective CA as the restriction of domains for which a CA is allowed to issue certificates is not considered. This is according to the current deployment of the Web PKI (cf. Section 3.1.3).

Attack surface measurements for real trust views

In the following, the reduction of the attack surface for the 64 trust views simulated based on the data sets collected in the user studies is presented. For all simulations,

the proposed parameter setting: $l_{max} = 0.95$, $l_{med} = 0.8$, $l_{min} = 0.6$, $maxF = 0.8$, $fix_{kl} = 3$, and $n = 10$ is used.

The calculation of the attack surface of the Web PKI in the system-centric setting is based on the number of 1,590 CAs observed by Durumeric et al. [17]. With the system-centric trust model for the Web PKI, all CAs are trusted for signing certificates for any domain, thus $AS = 1,590 \cdot dom$, which equals the attack surface computed with the original metric from [41].

The adapted metric enables the relative quantification of the reduction of the attack surface resulting from the use of CA-TMS with the above specified parameters. The relative attack surface for a trust view View is defined as:

$$AS_{rel}(\mathsf{View}) = \frac{AS(\mathsf{View})}{AS}.$$

Then, the reduction of the attack surface can be quantified as:

$$Red_{AS} = 1 - AS_{rel}(\mathsf{View}).$$

The distribution of the domains to dom_{max}, dom_{med}, and dom_{min} depends on the relying entity's preferences. Data about the distribution of security levels to domains is not available to us. The analysis is done for the three generalized cases where either l_{max}, l_{med} or l_{min} is assigned to all domains. We denote the respective relative attack surfaces with $AS_{rel}(\mathsf{View}, l_{max})$, $AS_{rel}(\mathsf{View}, l_{med})$ and $AS_{rel}(\mathsf{View}, l_{min})$. In these cases, the relative attack surface results as the quotient of CAs in the trust view that can issue certificates for the given security level, divided by the total number of CAs of the Web PKI. Further details about the data sets can be found in Tables 1 and 2 in the appendix.

For the 64 analyzed trust views we found $0 \leq AS_{rel}(\mathsf{View}, l_{max}) \leq 0.028$, $0 \leq AS_{rel}(\mathsf{View}, l_{med}) \leq 0.049$ and $0 \leq AS_{rel}(\mathsf{View}, l_{min}) \leq 0.057$.

The minimum numbers result from the least evolved trust views where nearly no CAs are trusted. In most cases this results from short histories and the related low number of observed hosts. However, considering the 48 trust views resulting from browsing histories with a minimum length of six months only slightly increases the minimal relative attack surfaces to 0, 0.001 and 0.001. The averages for these 48 trust views are $AS^{avg}_{rel}(\mathsf{View}, l_{max}) = 0.009$, $AS^{avg}_{rel}(\mathsf{View}, l_{med}) = 0.019$ and $AS^{avg}_{rel}(\mathsf{View}, l_{min}) = 0.026$.

This shows a reduction of the attack surface of at least 94.3%, even for the security level l_{min}. On average, a reduction of 97.4% is achieved in our data sets.

To generalize these results, the relative attack surface is evaluated depending on the number of observed hosts. The attack surfaces for the different security levels

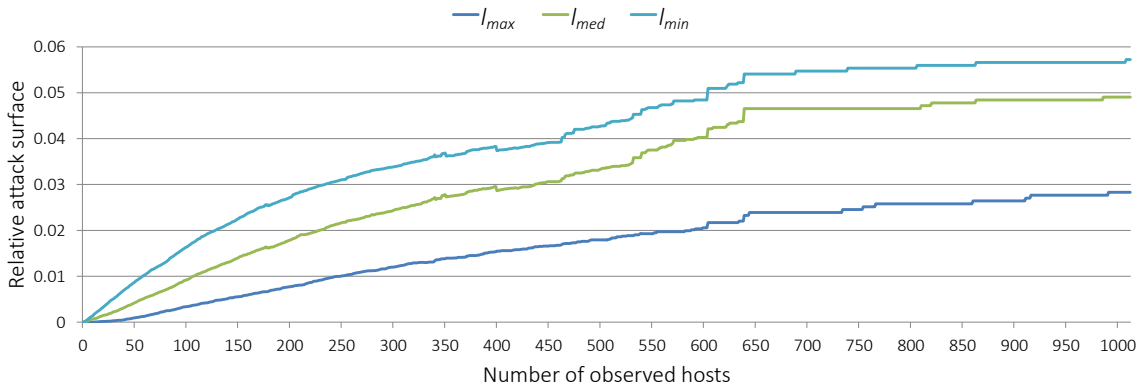


Figure 4.6: $AS_{rel}(\mathsf{View})$ for security levels l_{max}, l_{med} and l_{min}. (Real data.)

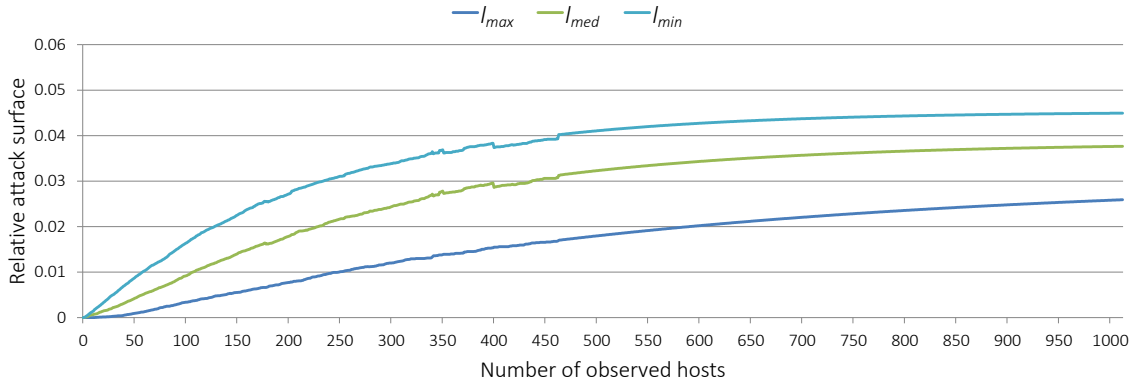


Figure 4.7: $AS_{rel}(\mathsf{View})$ for security levels l_{max}, l_{med} and l_{min}. (Extrapolated for host numbers larger than 466.)

are measured during the simulations after each observation of a new host. The resulting attack surfaces are then averaged over all simulated trust views. The results are shown in Figure 4.6. It shows an under proportional growth depending on the number of different hosts that are accessed via TLS secured connections.

The gaps within the graph result from trust views ending with the according number of observed hosts. Thus, the results for the high numbers of observed hosts depend on a low number of trust views. In our data sets, only 7 trust views contained more than 466 different hosts. Furthermore, the results for more than 639 hosts are based on a single trust view with 1013 hosts in total. This trust view also formed an upper bound for the number of observed CAs. Thus, the averaged results for high numbers of observed hosts are biased towards a larger attack surface. To have an estimate for the average relative attack surfaces, the data from the measured values below 466 observed hosts is extrapolated. This is depicted in Figure 4.7.

The extrapolation is done with an exponential estimator. The average rate of change was computed with a sliding window of size 50. The resulting curve was approximated with an exponential approximation function, which was used to extrapolate values of the attack surface for more than 466 hosts. Figure 4.7 shows that the relative attack surface on average stays below 0.05 even for the security level l_{min}.

The reduction of the attack surface comes at the cost of querying validation services whenever new CAs are observed or not enough trust experiences have previously been collected. In the next section we show, that the rate of reconfirmations is kept in an acceptable range and does not interfere browsing in practice.

4.4.3 Performance of CA-TMS

The performance of CA-TMS is evaluated in terms of the overhead induced by trust validation and continuous revocation monitoring. The evaluation is focused on noticeable delays during the use of CA-TMS, i.e. during browsing. As all computations are done locally, trust validation itself does not lead to a noticeable delay. However, requesting reconfirmations from validation services introduces communication overhead and delays, as page loading is blocked until the certificate has been reconfirmed. Thus, the performance of CA-TMS is evaluated based on the rate of reconfirmations. The performance of continuous revocation monitoring is evaluated in terms of daily OCSP requests.

Reconfirmation rates

The evolution of the 64 trust views is simulated based on the collected browsing histories. For all simulations, the proposed parameter setting: $l_{max} = 0.95$, $l_{med} = 0.8$, $l_{min} = 0.6$, $maxF = 0.8$, $fix_{kl} = 3$, and $n = 10$ is used. The reconfirmation rate is measured during the simulations after each observation of a new host. The reconfirmation rate is defined as:

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

with h_r denoting the number of hosts where a certificate reconfirmation was required and h denoting the total number of different hosts. The reconfirmation rates for the three different security levels l_{max}, l_{med} and l_{min} are measured independently. The results are averaged over all simulation runs of the 64 data sets.

Figure 4.8 shows the reconfirmation rates depending on the number of observed hosts. It shows how the percentage of hosts, for which a reconfirmation is required,

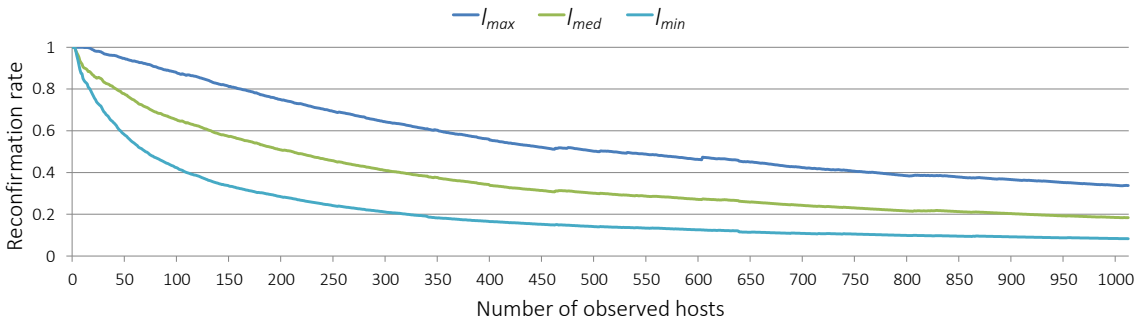


Figure 4.8: Reconfirmation rates: Percentage of observed hosts for which a reconfirmation is required, concerning the security levels l_{max}, l_{med} and l_{min}.

develops on average over the course of the evolution of trust views. The rates of reconfirmation continuously drop during the trust view evolution. Thus with each additional observed host, CA-TMS increasingly relies on local information. While for the first 50 observed hosts, the rates lie between 0.58 and 0.94 depending on the chosen security level, these drop to 0.34 and 0.08 for trust views with 1000 hosts.

While Figure 4.8 shows the continuous change of reconfirmation rates, Table 4.1 depicts the average rate for a block of 100 hosts after a certain number of hosts has already been observed. For example, given the security level l_{min}, 42% of the first one hundred hosts have to be reconfirmed, while this is only the case for 14.5% of the second one hundred accessed hosts.

The development of the reconfirmation rates shows the perceived improvement due to bootstrapping, even for incomplete histories. Already a small number of hosts leads to a significant drop in the reconfirmation rates. In Figure 4.9 it is depicted, when the reconfirmations occur during the evolution of a trust view with 1000 hosts. It shows, that most reconfirmations happen at the beginning. For the security level l_{min} 50% of all reconfirmations happen during the observation of the first tenth of hosts, and for the security level l_{max} 56% of all reconfirmations have

	Number of observed hosts									
	100	200	300	400	500	600	700	800	900	1000
l_{max}	0.880	0.620	0.431	0.308	0.274	0.267	0.190	0.110	0.220	0.090
l_{med}	0.654	0.364	0.214	0.138	0.134	0.125	0.070	0.030	0.100	0.020
l_{min}	0.424	0.145	0.064	0.031	0.042	0.047	0.007	0.030	0.040	0.010

Table 4.1: Average rates of reconfirmation for blocks of 100 hosts during the evolution of trust views for the security levels l_{max}, l_{med} and l_{min}.

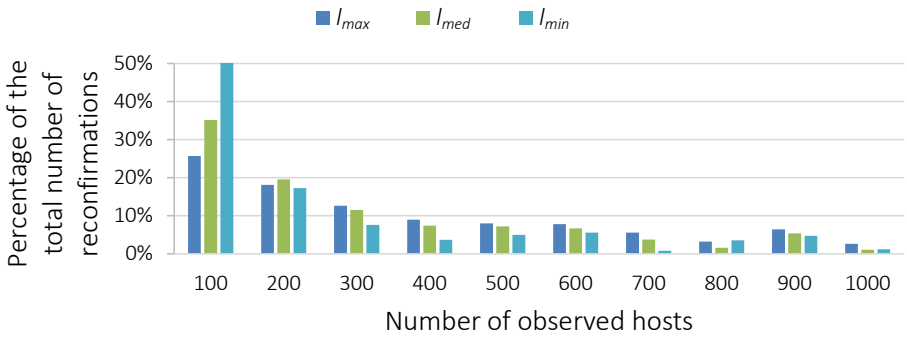


Figure 4.9: Percentage of the total number of reconfirmations during the evolution of trust views, concerning the security levels l_{max}, l_{med} and l_{min}.

happened during the observation of the first 30% of all hosts.

Repeatedly accessing a host does not lead to additional reconfirmations. Connections to new hosts on average make up for less than 1% of the total number of TLS connections (see also Table 4.2 below). Together with the reconfirmation rates presented above that means – depending on the security level – only between 0.27% and 0.69% of the TLS connections require a reconfirmation for an average user. Thus, delays induced by CA-TMS are hardly recognizable.

The reconfirmation rates can further be lowered with a reputation system as presented in Chapter 5. In the next section we evaluate the reconfirmation rates concerning known hosts.

Marginal reconfirmation rates

Once a trust view is bootstrapped, the certificates for the hosts are known and can be used during browsing without further checks or reconfirmations. However, these certificates are renewed every one to two years, whenever their validity ends or a new key is deployed on a server. Given that in most cases new certificates are obtained from the same CA [17], the reconfirmation rates for known hosts can be derived from the distribution of hosts to minimally, medium, and fully trustworthy CAs. We call these reconfirmation rates *marginal* reconfirmation rates. These are shown in Figure 4.10. For example, a trust view with 400 hosts only needs a second reconfirmation for 20%, 6% and 2% of the certificate changes of known hosts when browsing on the security levels l_{max}, l_{med} or l_{min}.

Assuming, the security levels are required for the same number of hosts, this results in an aggregated marginal reconfirmation rate of 9.3%. With a certificate renewal once a year, this implies one reconfirmation every 10 days. Thus CA-TMS only leads to delays in page loading occasionally. A relying entity which only

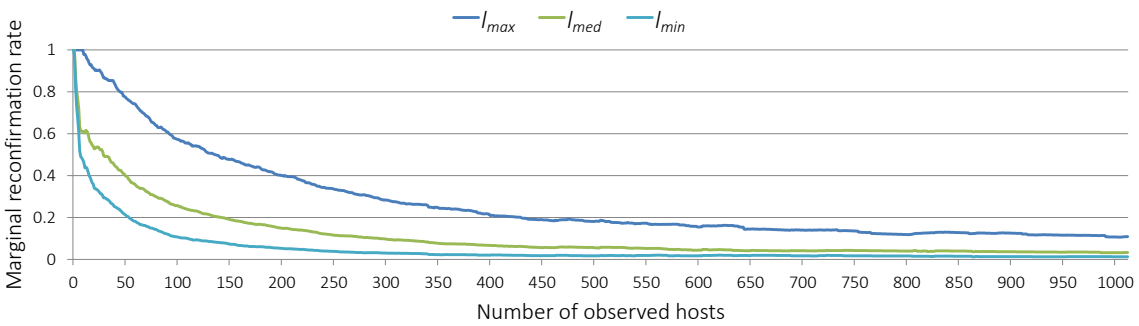


Figure 4.10: Marginal reconfirmation rates: Expected percentage of reconfirmations for certificate renewals of known hosts, for security levels l_{max}, l_{med} and l_{min}, depending on the number of different hosts associated to the trust view.

accesses a set of 100 different hosts via TLS secured connections is faced with higher reconfirmation rates. However, this is compensated by the lower total number of observed hosts. In this case, an aggregated marginal reconfirmation rate of 33% leads to one reconfirmation every 11 days.

The effects of stereotyping and fixing of the key legitimacy

The effects of stereotyping and fixing of the key legitimacy are evaluated by comparing the resulting reconfirmation rates during trust view evolution with and without these mechanisms. Over all security levels, enabling stereotyping and fixing of the key legitimacy reduces the reconfirmation rates by 50%-60%. However, for security levels l_{max} and l_{med} the main part of the reduction results from fixing the key legitimacy. Less than 10% of the reduction results from stereotyping. Thus, the limitation of the effects of stereotyping with parameter $maxF$ to low security applications is effective.

For security level l_{min} both mechanisms are equally effective. Equal parts of the reduction result from stereotyping and fixing the key legitimacy. The effects of the single mechanisms accumulate.

To sum up, using both strategies concurrently, significantly reduces the number of hosts for which a reconfirmation is required.

Continuous revocation monitoring

For the evaluation of the overhead induced by continuous revocation monitoring, we assume that the majority of certificates can be checked with OCSP. CRLs are only downloaded if OCSP is not available for a certificate. The evaluation focuses

	min	max	avg
Number of daily TLS connections	<1	407	61
Number of CAs	6	138	67
Number of hosts	2	1,013	214
Number of TLS connections	11	159,882	23,138

Table 4.2: Measurements for number of CAs, hosts and TLS connections from 64 browsing histories.

on the number of OCSP requests. In the current deployment, OCSP requests are sent during each TLS connection establishment. Thus, the number of daily OCSP requests equals the number of daily TLS connections. The numbers of daily TLS connections presented in Table 4.2 are averaged values over the total time periods covered by the respective browsing histories.

For continuous revocation monitoring, we propose a daily update of the revocation information. Thus, the daily amount of OCSP requests equals the number of CAs in the trust view. This is because requests for different hosts can be aggregated into one request when the same OCSP server is to be used (cf. Section 2.2.3), which in general is the case when certificates were issued by the same CA.

The measurements in Table 4.2 show that continuous revocation monitoring introduces additional OCSP checks if relying entities access few servers via TLS connections, and saves OCSP requests when many TLS connections are used. On average, comparable numbers for OCSP requests can be observed. Thus, continuous revocation monitoring does not introduce additional overhead. Even more, continuous revocation monitoring allows load balancing, as the requests are independent from the actual use and can be run in background.

4.5 Conclusion

In this chapter, the user-centric CA trust management system CA-TMS was presented. It maintains a minimal set of trusted CAs. By this user-centric management of trusted CAs, the attack surface of the Web PKI is reduced. With the presented parameter setting, CA-TMS achieves a reduction of the attack surface of more than 95%. This prevents attacks induced by CA failures and compromises. By making use of validation services, a relying entity's trust view is dynamically adapted to the individual needs, while user interaction can be kept to a minimum. Furthermore, CA-TMS assigns different ratings to each CA, such that trust decisions can be made depending on the context. Thus, the risk of relying on a fraudulent certificate can be

governed by the assignment of adequate security levels to applications. This enables more restrictive trust decisions for critical applications like e-banking, where security is more important and less restrictive rules for less security critical applications. These rules can be adapted to the relying entity's risk profile.

The provision problem of revocation information was solved with continuous revocation monitoring, which decouples revocation checking from the actual use of certificates.

Additionally, an implementation of CA-TMS has been presented. It integrates currently available solutions for certificate reconfirmation such as certificate notaries, and allows the flexible addition of more information sources to further improve reconfirmation and data collection processes. The core functionality of CA-TMS can be implemented locally on top of the existing infrastructure without its alteration.

The performance evaluation showed that the need for reconfirmations fades out the more local experiences are collected. This continuously reduces the overhead and possible delays. Once the system is bootstrapped, reconfirmations on average occur once every few days. A reconfirmation is only required for 0.27% to 0.69% of the TLS connection establishments. In summary, this shows that the use of CA-TMS does not interfere normal browsing. The performance analysis also showed that continuous revocation monitoring can be implemented without additional overhead.

CA-TMS so far protects relying entities from CA compromises concerning CAs not rated trustworthy in their trust views. Exploiting the individuality of these trust views enables a detection mechanism for fraudulent certificates, which we present in Chapter 5. Combined with a push service, this mechanism extends the protection capabilities of CA-TMS to also protect from CAs which suddenly change their behavior.

5 Service providers for CA-TMS

In Chapter 4, it was shown how to protect relying entities from fraudulent certificates issued by CAs that are not part of their trust view. A bootstrapping of the trust view based on a relying entity's browser history was proposed and it was shown that the presented system provides good performance once the trust view is bootstrapped.

However, problems evolve when no browser history is available for bootstrapping or when CAs that are already considered to be trustworthy change their behavior and become untrustworthy. The lack of a browser history implies that the trust view needs to be bootstrapped on the fly during browsing, which then is performed over the course of several months. This is problematic because the certificate checks during bootstrapping lead to interruptions and delayed page loading. Furthermore, the availability of external validation services is critical whenever a new service is to be accessed. The problem with behavioral changes of CAs is, that relying entities that consider a certain CA as trustworthy will accept certificates issued by that CA without further checks. If such a CA suddenly starts to issue fraudulent certificates, e.g., because of a compromise, this stays undetected by the group of relying entities that trusts this CA.

In this chapter, we solve these two problems by extending CA-TMS with online service providers to realize a centralized reputation system for CA trust scores and a push service to warn relying entities. After a functional description, the services are evaluated.

In Sections 5.1 - 5.3 we describe the functionality of the service providers. The bootstrapping problem is solved with the reputation system. It makes the knowledge of other relying entities available to a relying entity whenever his own experiences are insufficient for decision making. With this mechanism, we speed up the bootstrapping process. We describe the trust view selection and trust aggregation mechanisms and show how protection against malicious recommenders is realized. The second problem is solved with the push service. The service providers monitor the CA trust scores collected within the reputation system. Upon detection of a CA becoming untrustworthy, this information is pushed to all clients whose trust view contains

the CA in question. We describe the detection mechanisms and how the pushed information is processed on the client side.

In Section 5.4 the presented services are evaluated. First, we extend our attacker model from Chapter 3 with additional attack vectors induced by the incorporation of trust information provided by other relying entities, e.g., such as sybil attacks. Then, we evaluate the security of the proposed services based on the extended attacker model. The functionality and the performance is evaluated based on simulations using real world browsing histories. We simulate the bootstrapping process of trust views with and without the reputation system. The timings required for bootstrapping as well as data loads and traffic overheads are compared between the two settings. With the metric presented in Chapter 4 for the measurement of the attack surface, we evaluate the effects of the inclusion of the reputation system on the overall attack surface. The push service is evaluated in terms of an attacker's success probability for different attack scenarios. Section 5.5 concludes the chapter.

Parts of the contributions of this chapter were published in [B2], which covers the concept of the reputation system. This chapter extends the published contributions by the sections concerned with the push service and its evaluation. Additionally, the performance evaluation for the reputation system was added.

5.1 Architecture and system model

To realize the reputation system and the push service to warn relying entities about behavioral changes of CAs, we introduce online service providers to CA-TMS. This is realized in a centralized architecture, which for scalability reasons can be extended to a network of service providers. The relying entities register at the service provider and upload their trust views. Note that the authentication of a single service provider is not considered a problem. E.g., its certificate may be hard coded into the client software. Due to the registration, relying entities can be re-identified when accessing the provided services.

Model For a common understanding, we extend the system model from Chapter 4 with service providers. Recall that there exists an entity $\mathcal{E}_1$ with trust view View and another entity $\mathcal{E}_2$. $\mathcal{E}_1$ establishes a TLS connection to $\mathcal{E}_2$ and needs to decide whether the connection is trustworthy or not.

Now, other entities $\mathcal{U}_1, \ldots, \mathcal{U}_n$ (other Internet users) with trust views $\mathsf{View}_1, \ldots, \mathsf{View}_n$ and a network of service providers $\mathtt{SP}_1, \ldots, \mathtt{SP}_m$ are additionally included in the model. The service providers are assumed to have pre-established trust relation-

ships and are able to communicate securely, i.e. their keys are exchanged using an out of band channel. The network of service providers does not need to be complete, i.e., a service provider is not required to trust any other service provider. We assume that $\mathcal{E}_1$ and $\mathcal{U}_1, \ldots, \mathcal{U}_n$ have registered at $\mathtt{SP}_1$ and uploaded their trust views to the database of $\mathtt{SP}_1$. 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 $\mathtt{SP}_i$'s database. Figure 5.1 depicts the architecture.

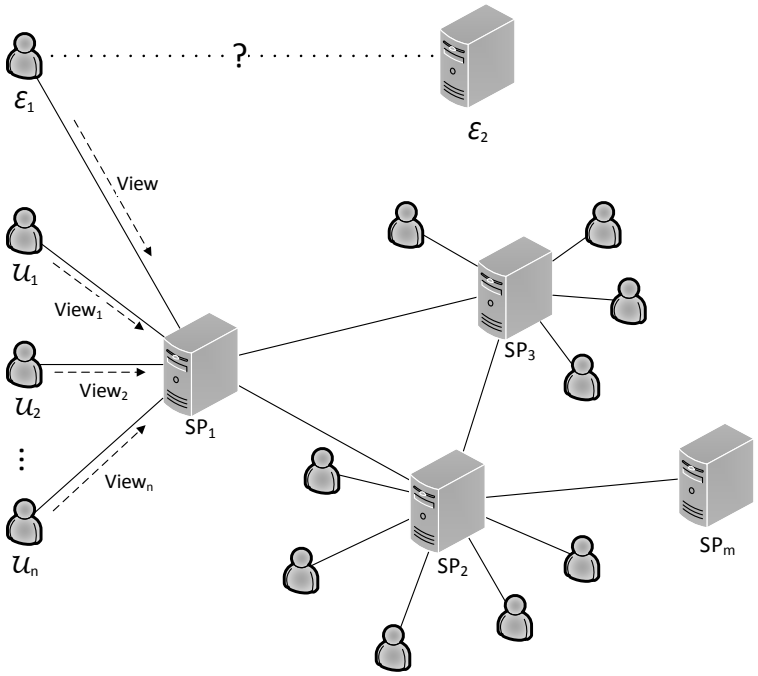


Figure 5.1: Service provider architecture

5.2 Reputation system for CA trust management

In Chapter 4, we have described how the trust view is established incrementally by querying validation services for any certificates as long as not enough locally collected information is available. In this section it is shown how to integrate an additional reputation system to increase the amount of data that decisions are based on from local experiences to aggregated opinions from people with similar browsing behavior. The service provider realizing the reputation system aggregates the opinions from a set of trust views in his database to a recommendation and provides it to a requesting entity. While doing this, the trust views are kept minimal in order to adhere to

the principle of least privileges, which is currently not followed in the Web PKI [17]. Therefore, an entity's requirements concerning CAs have to be considered during the aggregation of the recommended issuer trust. Otherwise, experiences collected for application uses completely irrelevant to the requesting entity might lead to unnecessarily high trust values in this entity's trust view. Furthermore, the reputation system is only queried after the relevance of a CA for an entity was approved due to the CA being part of a reconfirmed certification path (cf. Section 5.2.1). The purpose of the reputation system is to improve the bootstrapping of the trust view and to evolve an entity's trust view towards a stable state. When reaching a stable state, CA-TMS may work mainly autonomously, even for entities that only collect few own experiences.

The remainder of this section is organized as follows. First, the basic functionality of the reputation system is described in Section 5.2.1. Afterward, in Section 5.2.2, we present a strategy to select trust views for the computation of the recommended issuer trust. Then, it is shown how the selection of trust views is dynamically adapted to the maturity of an entity's trust view. In the case of insufficient information on the reputation system's side, a service provider handover may be performed as described in Section 5.2.3. Finally in Section 5.2.4, we discuss privacy aspects that arise from uploading personal trust views to the reputation system.

5.2.1 Functionality

The reputation system provides recommendations for the issuer trust assigned to a CA. Whenever $\mathcal{E}_1$ updates his trust view View with a new trust assessment TA for CA $\mathtt{CA}$ with public key pk, $\mathcal{E}_1$ requests a recommendation for the associated issuer trust from $\mathtt{SP}_1$. $\mathtt{SP}_1$ aggregates the correspondent opinions for $\mathtt{CA}$ from j different trust views by applying the conflict aware fusion operator cFUSION (cf. Definition 2.6). In case $\mathtt{SP}_1$ does not have information about $\mathtt{CA}$ with key pk, $\mathtt{SP}_1$ forwards the request to other service providers he trusts.

The process is the following:

1. $\mathcal{E}_1$ establishes a TLS connection to $\mathtt{SP}_1$ and authenticates itself (e.g., by using a user name and password).

2. $\mathcal{E}_1$ sends the pair $(\mathsf{pk}, \mathtt{CA})$ to $\mathtt{SP}_1$ using the secure connection.

3. Depending on View, $\mathtt{SP}_1$ selects $j \geq 0$ trust views $\mathsf{View}_1, ..., \mathsf{View}_j$ from its database (see Section 5.2.2 for the selection strategy).

4. If $j > 0$ do
   a) For $1 \leq i \leq j$ $\mathtt{SP}_1$ extracts $o^{ca}_{it,i}$ and $o^{ee}_{it,i}$ for $(\mathsf{pk}, \mathtt{CA})$ from View_i.
   b) $\mathtt{SP}_1$ aggregates the opinions on the issuer trust with the cFUSION operator: $\tilde{o}^{ca}_{it} = \widehat{\oplus}_c(o^{ca}_{it,1}, ..., o^{ca}_{it,j})$ and $\tilde{o}^{ee}_{it} = \widehat{\oplus}_c(o^{ee}_{it,1}, ..., o^{ee}_{it,j})$.
5. If $j = 0$ (i.e., $\mathtt{SP}_1$ has no information for $(\mathsf{pk}, \mathtt{CA})$) $\mathtt{SP}_1$ forwards the request to another service provider it trusts. The other service provider responds with a recommendation $(\tilde{o}^{ca}_{it}, \tilde{o}^{ee}_{it})$ or with `unknown`. This step may be repeated or run in parallel for several service providers.
6. $\mathtt{SP}_1$ responds to $\mathcal{E}_1$ with either the aggregated issuer trust opinions $(\tilde{o}^{ca}_{it}, \tilde{o}^{ee}_{it})$ or, if no recommendation is available, with `unknown`.
7. $\mathcal{E}_1$ integrates the recommendation into View.

Before describing in detail how $\mathtt{SP}_1$ selects trust views for the aggregation and how to aggregate opinions to a single recommendation, we recap how the reputation system is integrated into CA-TMS on the side of $\mathcal{E}_1$. As described in Chapter 4, $\mathcal{E}_1$ requests a recommendation for newly observed CAs as part of the trust view update. We shortly recap Step 3 of the trust view update algorithm presented in Section 4.1.6:

3. *(optional)* If $(R = \mathtt{trusted}) \wedge (v = \mathtt{true})$ then $\forall \mathsf{TA}_i \in TL$ do:
   a) Request $\mathtt{RS}(\mathsf{pk}_i, \mathtt{CA}_i) = (\tilde{o}^{ca}_{it,i}, \tilde{o}^{ee}_{it,i})$ from $\mathtt{RS}$
   b) If $\mathtt{RS}$ did not return `unknown` do:
      If $(i < n - 1)$ set $o^{ca}_{it,i} = (0.5, 0, E(\tilde{o}^{ca}_{it,i}))$, else set $o^{ee}_{it,i} = (0.5, 0, E(\tilde{o}^{ee}_{it,i}))$.

The trust view update algorithm ensures that two preliminaries are fulfilled before the reputation system $\mathtt{RS}$ is queried and recommendations are integrated into $\mathcal{E}_1$'s trust view View. First the outcome of trust validation for the certification path in question is required to be `trusted`: $(R = \mathtt{trusted})$. This ensures, that the CAs for which a recommendation is requested are indeed relevant to $\mathcal{E}_1$ as parts of a legitimate certification path. Second, the certification path has to be reconfirmed by validation services: $(v = \mathtt{true})$. This second condition ensures, that indeed external information is required and not enough local information was already available for trust validation. Thus, the principle of least privileges is realized as stated above.

$\mathcal{E}_1$ integrates the recommendations into View by setting the initial trust value f of the according issuer trust opinions in View to the expectation of the recommended issuer trust opinions $(\tilde{o}^{ca}_{it,i}, \tilde{o}^{ee}_{it,i})$. With this, it is ensured that $\mathcal{E}_1$ only relies on external

information as long as local information is missing. Recall that the influence of f on the expectation of an opinion ceases with a growing amount of experiences. Second, it prevents circular dependencies between locally collected experiences and the recommendations, thus preventing a bias of the system.

5.2.2 Trust view selection and trust aggregation

First the selection and aggregation strategy is presented. Afterward, it is shown how to apply clustering for pre-computation and efficiency improvements.

Trust view similarity weighting and cut off

The aggregation of the recommended issuer trust should consider an entity's individual requirements. This cannot be achieved by simply averaging the respective opinions over all trust views in the service provider's database. The recommendation should be based on the trust views of entities that have comparable requirements as the requesting entity, namely entities with similar browsing behavior and similar security requirements. Because CAs mostly issue certificates for a limited set of domains [41] and because trust views depend on the subjective browsing behavior as shown in Chapter 3, we deduce that trust views reflect an entity's requirements in respect to the relevance of CAs. The correctness of this assumption will be shown in Section 5.4.3.

Thus, the input opinions for aggregation are weighted according to the similarity between the trust views from which they originate and the trust view of the requesting entity. Furthermore, all opinions with a similarity below a certain lower bound $b \in \mathbb{R}$ are cut off. Similarity of trust views can be measured with the Jaccard similarity index [16].

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 trust assessments contained within trust views, the Jaccard similarity index can be applied to trust views as follows. For this, we consider two trust assessments 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 trust assessments contained in $\mathsf{View}_i,\ i \in \{1, 2\}$ and n is the number of trust assessments shared by View_1 and View_2. Note that the inclusion of certificates into the computation is omitted due to privacy issues (cf. Section 5.2.4). In Section 5.4.3 we evaluate the suitability of the Jaccard similarity and show that it fits the requirements.

Selection and aggregation To compute the opinions for the recommendation $\mathtt{SP}_1$ retrieves all trust views from its database that contain a trust assessment for $(\mathsf{pk}, \mathtt{CA})$ specified in the request by $\mathcal{E}_1$. Let $TV_{(\mathsf{pk},\mathtt{CA})}$ be the set of these trust views. Then, using $\mathcal{E}_1$'s trust view View, $\mathtt{SP}_1$ computes for each $\mathsf{View}_i \in TV_{(\mathsf{pk},\mathtt{CA})}$ the corresponding weight $w_i = J(\mathsf{View}, \mathsf{View}_i)$. Afterward, any trust view $\mathsf{View_i}$ with $w_i \leq b$ is discarded from $TV_{(\mathsf{pk},\mathtt{CA})}$.

From the remaining j trust views, $\mathtt{SP}_1$ extracts the opinions on issuer trust $o^{ca}_{it,i}$ and $o^{ee}_{it,i}$ for $(\mathsf{pk}, \mathtt{CA})$ and aggregates them using the cFUSION operator (cf. Definition 2.6) using the corresponding weights w_i. Thus, opinions originating from trust views that are more similar to View have a stronger influence on the aggregated recommendation than opinions from less similar trust views. Furthermore, if there exist conflicting opinions on the trustworthiness of the CA, the cFUSION operator handles this by lowering the certainty of the result.

Trust views with weights below the bound b are cut off in order to prevent trust views that are highly dissimilar to View from being taken into account. This is done because the cFUSION operator only considers weights relatively. In case that solely trust views with a low Jaccard similarity are found, this would result in relatively high weights. Cutting off those trust views may come at the cost of not finding adequate trust views but prevents the recommendation of high issuer trusts, if the importance of a CA to an entity is not plausible. Besides that, the similarity weighting and cut off strategy provides protection against Sybil attacks which we discuss in Section 5.4.2.

Because CA-TMS incrementally learns an entity's trust view over time, the reflection of the browsing behavior is limited during the bootstrapping phase. In order not to cut off trust views just because of the lack of local information, the bound b is dynamically adjusted to the maturity of a trust view. This is described in the following.

Dynamic adaptation of the bound b

As soon as a trust view has a sufficient maturity, the Jaccard similarity can be used to identify trust views of other entities with similar browsing behavior. Yet as mentioned above, during the bootstrapping phase no or only limited information is available about an entity's own trust view and thus its browsing behavior. This has two effects. First, the significance of the similarity to other trust views is limited. Second, it results into low similarity values to evolved trust views that potentially contain information about newly observed CAs, leading to their exclusion from the aggregation of a recommendation if the bound b for similarity cut off is not adequately set.

To overcome these problems, b is dynamically increased during the bootstrapping of a trust view. This dynamic adaptation is chosen such that trust views resulting from identical browsing behavior but with high maturity are not excluded from recommendation aggregation.

A low value of b at the beginning subsequently leads to the inclusion of a broad variety of trust views resulting in generalized recommendations. By the dynamic adaptation, the choice is increasingly focused on trust views from the same user group, thus providing increasingly individualized recommendations. In parallel, similarity weighting ensures, that already collected information is mirrored in the recommendation aggregation and the input opinions from the more similar trust views have stronger influence on the recommendation than opinions from less similar ones.

The adaptation function for b is derived from the set of 64 real world browsing histories also used for the evaluations. From each of the histories the final state of the associated trust view is derived and its evolution is simulated. During the simulations, the Jaccard similarity to the trust view's final state is computed after each observation of a new host. The development of these similarities for each trust view is displayed in Figure 5.2. For each number of observed hosts, the minimal similarity value is extracted. This minimum forms the lower bound of similarities for all sets. From the data series formed by the minimal similarity values, the logarithmic approximate function is computed as:

$$f(h) = 0.2223 \cdot ln(h) - 0.5036,$$

where h represents the number of observed hosts. $f(h)$ is displayed as the blue curve in Figure 5.2. The respective coefficient of determination is $R^2 = 0.9706$. The approximate function is used to dynamically compute b depending on the number of observed hosts. We set $b = f(h)$ for $h < 352$. The dynamic adaptation is stopped at $h = 352$, where $f(h)$ approaches 0.8. Afterward, the bound is fixed at $b = 0.8$ to

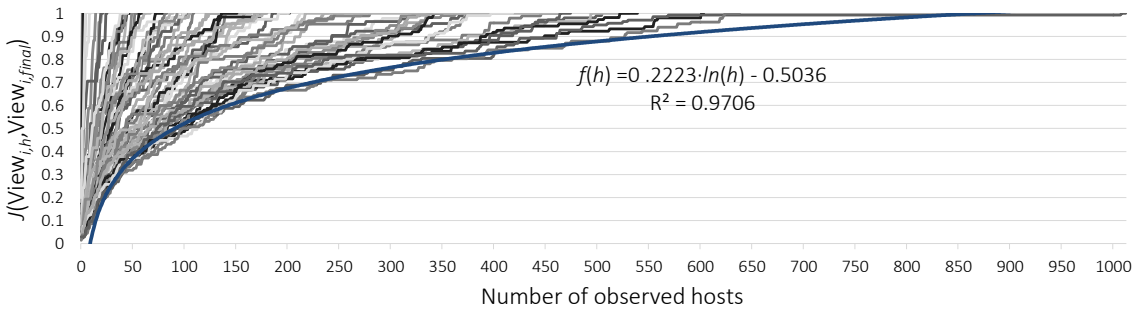


Figure 5.2: Evolution of Jaccard similarities of 64 trust views. The similarities are calculated between the final trust view state and the trust view after observation of x hosts.

ensure that additional CAs are accepted within the trust views selected for opinion aggregation.

Trust view similarity clustering

A drawback of trust view similarity weighting and cut off 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 $\mathtt{SP}_1$'s database that contain a trust assessment for $(\mathsf{pk}, \mathtt{CA})$. This can be done more efficiently by clustering the trust views and then performing a local search within the cluster to which View is assigned.

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

K-means clustering With K-means clustering and the Jaccard similarity index for trust views, κ clusters of similar trust views can be built. According to [29], 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 the 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 [35], 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 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.

Initially, the parameter κ, which steers the number of clusters and thus their sizes, is chosen such that searching for the right cluster and searching within a cluster is balanced. Thus, given the service provider's database contains n trust views, we set $\kappa = \sqrt{n}$.

5.2.3 Service provider handover

In case the service provider $\mathtt{SP}_1$ has no trust views to compute a recommendation for $(\mathsf{pk}, \mathtt{CA})$ – either there is no trust view with a trust assessment for $(\mathsf{pk}, \mathtt{CA})$ at all or none that meets the minimal similarity constraint given by the bound b – $\mathtt{SP}_1$ can request a recommendation from other service providers $\mathtt{SP}_2, \ldots, \mathtt{SP}_m$ it trusts.

$\mathtt{SP}_1$ queries $\mathtt{SP}_2, \ldots, \mathtt{SP}_m$ for their recommendation. If more than one answers with a recommendation $(\tilde{o}_{it}^{ca}, \tilde{o}_{it}^{ee})$, 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 $\mathtt{SP}_2, \ldots, \mathtt{SP}_m$ respond with `unknown`, $\mathtt{SP}_1$ also responds with `unknown` to the requesting entity $\mathcal{E}_1$.

In order to enable $\mathtt{SP}_2, \ldots, \mathtt{SP}_m$ to locally perform the aggregation of opinions to a recommendation as described in Section 5.2.2, $\mathtt{SP}_1$ hands over $\mathcal{E}_1$'s trust view. To protect $\mathcal{E}_1$'s privacy, the trust view is shortened by all end entity certificates.

5.2.4 Privacy aware data collection

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

In this case privacy protection has two aspects. First the privacy sensitive data revealed to the service provider is to be kept minimal depending on the necessity. The service provider only requires the trust assessments in plain to be able to realize the reputation system as well as the push service presented in the following section. To protect a user's privacy, the parts of the trust view that reveal which services a user consumes (namely the set of trusted and untrusted end entity certificates) are stored encrypted with a user specific key. The privacy criticality of the information associated with the trust assessments themselves is limited because CAs in general sign certificates for arbitrary web services.

The second aspect of privacy protection concerns the information which is revealed to other potentially untrusted third parties, which refers to the recommendations. Recommendations are aggregated knowledge of several of the service provider's users. The recommendations are furthermore not linked to the source of information. Thus, a recommendation reveals the trust in a single CA in an anonymous fashion and therefore are not considered privacy critical. Trust views are never revealed as a whole to the public.

5.3 Push service for behavioral changes of CAs

CA-TMS collects experiences concerning past behavior of CAs and uses this information to estimate the CA's trustworthiness concerning future transactions. In Section 5.2 a reputation system was described, that enables to speed up information collection, finally enabling autonomous decision making by CA-TMS.

In this section we elaborate on the issue of behavioral changes of CAs and describe a detection mechanism which enables a push service to inform relying entities about such behavioral changes. Behavioral changes may lead to false decisions, because then, collected experiences do not reflect the real behavior anymore at the time of decision making. The slow adaptation to behavioral changes is a common property of many computational trust models [67]. In the case of CA-TMS, only a sudden change from trustworthy to untrustworthy is security critical. A change from untrustworthy to trustworthy is not a problem. In this case the relying entity queries validation

services to reconfirm certificates issued by the CA and finally learns the changed behavior.

The problem with the first case is, that relying entities that already consider a CA as trustworthy will accept certificates issued by that CA without further checks. And if this CA suddenly starts to issue fraudulent certificates, e.g. because of a compromise, this stays undetected. Furthermore, relying entities do not query the reputation system regularly and update their trust scores accordingly. This would weaken the individuality of the trust views. Thus, a relying entity might not recognize the change even if others already detected it. Even if the misbehavior is later detected and negative experiences are collected in retrospect, the attack already occurred and because of the slow trust adaptation it is still possible, that further wrong decisions are made.

Preventing relying entities from attacks therefore requires immediate measures. This is solved by integrating the push service that informs users about behavioral changes of CAs contained in their trust views. To overcome the drawback of slow trust adaptation, we follow the proposal from [67]. Relying entities that are warned about a CA directly suspend it and prune the CA's performance history completely. This way, the relying entity learns the actual trustworthiness anew and does not rely on outdated information.

In the following Section 5.3.1, we describe how behavioral changes are detected by the service provider in collaboration with its clients. Section 5.3.2 presents how warnings are pushed to relying entities. In Section 5.3.3 it is described how relying entities process such push messages.

5.3.1 Report functionality for behavioral changes

In order to detect untrustworthy CAs automatically, the fact that trust views differ between relying entities is exploited. Let the target group be the group of relying entities which are attacked, i.e., which use the service for which the attacker obtained a fraudulent certificate. Our mechanism relies on the assumption, that a certain fraction of the target group does not trust the issuer of the fraudulent certificate. If this certificate is delivered to such an entity, it will subsequently be reconfirmed with validation services and be evaluated as untrustworthy. If this happens, the entity reports the certificate to the service provider as suspicious. The service provider validates the correctness of the report and in case it is confirmed, it pushes warnings to all its clients that have the issuing CA in their trust views. The detection capability of this mechanism is evaluated in Section 5.4.4.

Integration of the report functionality on the relying entity's side

The report functionality is integrated into the trust validation algorithm (cf. Section 4.1.5) as Step 3.l and is executed, whenever the trust validation algorithm evaluates a certification path as `untrusted`. The report protocol is the following:

Input:

- The certification path $p = (C_1, ..., C_n)$.
- The URL *url* of $\mathcal{E}_2$ from which p was obtained.

Output: Whether the reporting was successful or not.
The protocol proceeds as follows:

1. $\mathcal{E}_1$ establishes a TLS connection to $\texttt{SP}_1$ and authenticates itself (e.g., by using a user name and password).
2. $\mathcal{E}_1$ sends the tuple (p, url) to $\texttt{SP}_1$ using the secure connection.
3. $\texttt{SP}_1$ confirms the report to $\mathcal{E}_1$ with a success message.

If the protocol does not finalize with the success message, $\mathcal{E}_1$ puts the report (p, url) into a queue and retries until the report can be sent.

Processing untrusted certificate reports on the service provider's side

To prevent false warnings, service providers validate each untrusted certificate report for correctness before pushing a warning to its clients. Note that costly reconfirmations are acceptable, as this is done by a single service provider and not by a huge amount of relying entities. Furthermore, delays of several seconds or even minutes outweigh false warnings which lead to the erroneous suspending of CAs and subsequently to non-justified page loading delays on the relying entity's side.

When the service provider obtains a report (p, url) about an untrusted certificate from one of its clients, it evaluates its correctness as described in the following:

Input:

- An untrusted certificate report $(p = (C_1, ..., C_n), url)$.
- A list of validation services $VS = (\texttt{VS}_1, ..., \texttt{VS}_j)$ with outputs $R_i = \texttt{VS}_i(C) \in \{\texttt{trusted}, \texttt{untrusted}, \texttt{unknown}\}, 1 \leq i \leq j$ on input of a certificate C.

Output: $V \in \{\texttt{valid}, \texttt{invalid}, \texttt{unknown}\}$.
The protocol proceeds as follows:

1. $\texttt{SP}_1$ performs standard path validation (including revocation checking) on the certification path p.
2. If path validation fails then $V \leftarrow \texttt{invalid}$
3. If path validation succeeds then
   a) For $1 \leq i \leq j$ query validation service $\texttt{VS}_i$ for C_n and set $R_i = \texttt{VS}_i(C_n)$.
   b) If $\exists i \in \{1, ..., j\}$ with $R_i = \texttt{untrusted}$ then $V \leftarrow \texttt{valid}$
   c) Else if $\exists i \in \{1, ..., j\}$ with $R_i = \texttt{trusted}$ then $V \leftarrow \texttt{invalid}$
   d) Else $V \leftarrow \texttt{unknown}$
4. Return V

If the report validation outputs `valid`, $\texttt{SP}_1$ pushes a warning to its clients and to the other service providers. If the report is `invalid`, it is discarded. In case the validity of a report is `unknown` it is queued and retried later. Note that the unknown case is very uncommon, when combining different types of validation services based on different principles such as multi path probing and notaries that cache valid certificates.

5.3.2 Pushing CA warnings to relying entities

Pushing warnings to clients is performed depending on the necessity. If the untrusted certificate report is valid, first the report is broadcast to all other service providers. Furthermore, for each valid report $(p = (C_1, ..., C_n), url)$ (either obtained by a client or another trusted service provider), $\texttt{SP}_1$ searches its database for trust views that contain a trust assessment for the key `pk` of CA `CA` certified in C_{n-1}. For each such trust view, the CA warning $(\texttt{pk}, \texttt{CA}, C_n)$ is pushed to the push address of the according relying entity. The push address is defined, when a relying entity registers at the service provider. The push service can be realized by implementing a push server, e.g. using the SimplePush API [160] specified by Mozilla, to which relying entities connect when they are online. Another possibility would be to use existing infrastructure and realize the push service via email. In this case, the client software would connect to the relying entity's email provider and monitor the inbox for push emails by its service provider. The latter option saves resources on the service providers side, while the email providers' infrastructures are in general already

available and are designed to handle such requests, as common email clients such as Thunderbird or Outlook synchronize the inbox every few minutes anyway.

5.3.3 Processing CA warnings

On receipt of an CA warning $(\mathsf{pk}, \mathtt{CA}, C_n)$ pushed by the service provider, the relying entity directly suspends CA, the issuer of C_n. This is done by resetting the opinions in the respective trust assessment to the initialization values. Furthermore, a negative experience is collected for CA. The untrusted certificate C_n is added to the relying entity's list of untrusted certificates. The CA's certificate is not marked as untrusted. With this, the relying entity learns the actual trustworthiness of the CA anew, and even allows the CA to become trustworthy again. This is important, as the issuance of a single fraudulent certificate may result from a temporary error. On the other hand, no certificates issued by the CA will be trusted without a previous reconfirmation until the relying entity learned that the CA is trustworthy again. This has the effect, that a relying entity is also protected from further fraudulent certificates issued by the same CA, even if they are not reported through the service provider. Certificates that are already in the list of trusted certificates are kept, as the current behavior the CA has no retroactive effects.

The whole process is transparent to the relying entity and runs in background except for the case, that C_n is found in the list of trusted certificates. In this case, an attack has already happened, and the relying entity must be informed about the incident such that it can take adequate measures, as e.g., blocking his bank account, to prevent further damage.

5.4 Evaluation

In this section we evaluate the reputation system and the push service. First, in Section 5.4.1 the attacker model from Chapter 3 is extended with additional attack vectors induced by the use of the reputation system. Then, in Section 5.4.2 we evaluate the security of the proposed system based on the extended attacker model. Afterward, in Section 5.4.3 the functionality and the performance of the reputation system is evaluated based on simulations using real world browsing histories. It is shown, that the Jaccard similarity index is adequate for weighting and trust view pre-selection within the reputation system. Then, we show, that the reputation system speeds up the data collection and improves the bootstrapping process. We simulate the bootstrapping process of trust views with and without the reputation system. The timings required for bootstrapping as well as data loads and traffic

overheads are compared between the two settings. With the metric presented in Chapter 4 for the measurement of the attack surface, we evaluate the effects of the inclusion of the reputation system on the overall attack surface. Finally, in Section 5.4.4, the push service is evaluated in terms of an attacker's success probability for different attack scenarios.

5.4.1 Attacker model

The security analysis is based on the attacker model presented in Section 3.1.2. Recall that attacker $\mathcal{A}$ is an active man-in-the-middle attacker on the connection between relying entity $\mathcal{E}_1$ and web server $\mathcal{E}_2$. $\mathcal{A}$ can generate certificates that are signed by a CA of the Web PKI. The systems of $\mathcal{E}_1$ and $\mathcal{E}_2$ are assumed not to be compromised, nor can $\mathcal{A}$ break the employed cryptographic algorithms. Also, the validation services are assumed to function correctly.

The incorporation of a reputation system RS into CA-TMS induces additional attack vectors. The attacker has the following additional capabilities and limitations.

Additional capabilities $\mathcal{A}$ has all capabilities of regular entities. For example, the attacker can register at a service provider and upload trust views to RS. Furthermore, $\mathcal{A}$ may compromise some of the other users $\mathcal{U}_1, \ldots, \mathcal{U}_n$ of RS.

Additional Limitations RS itself is assumed not to be compromised, i.e., $\mathcal{A}$ is unable to arbitrarily manipulate the database of RS or its computation processes. Even further, it is assumed that the communication between $\mathcal{E}_1$ and RS is secure. Intrusion detection and attacks on user systems are separate fields of security research and are out of scope of this thesis.

5.4.2 Attacks against CA-TMS

This section is focused on attacks against specific components of CA-TMS. It extends the evaluation of Chapter 4 by attacks aiming towards manipulating the external reputation system. This ultimately leads to a manipulation of $\mathcal{E}_1$'s local trust information. Following from the attacker model, the intention behind these attacks is to prevent detection when $\mathcal{A}$ attacks $\mathcal{E}_1$'s communication employing a fraudulent certificate. Recall that RS is only queried when a new trust assessment is initialized after the legitimate use of the CA's key has at least been reconfirmed once. Thus, RS cannot be employed by $\mathcal{A}$ to introduce additional trust assessments into $\mathcal{E}_1$'s trust view View. As in Chapter 4, the analysis is based on the framework from [31]

and identifies the attack vectors according to the separation into formulation, calculation, and dissemination components of CA-TMS. We shortly recapitulate their meaning and explain where the incorporation of RS adds new components.

Formulation resembles the reputation metric and sources of input. RS extends CA-TMS by an indirect information source regarding a CA's reputation. The information source for RS are trust views uploaded by its users. Thus, $\mathcal{A}$ can positively or negatively influence the trust in a CA by influencing the recommendations served by RS. RS itself may be influenced via uploading manipulated trust views.

Calculation concerns the algorithms that derive trust from the input information. With RS, non-local calculations concerning the recommendations are added to CA-TMS. The recommendations are calculated deterministically and centrally by RS. Thus, as RS is assumed not to be compromised, $\mathcal{A}$ cannot influence the calculation other than by injecting manipulated input data into the database of RS.

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

Attack vectors

Following from the analysis, the possibilities to manipulate and influence the trust evaluation of CA-TMS under the given assumptions is limited to injecting false input data. We now explain the attack vectors $\mathcal{A}$ might use to inject false data indirectly via the RS, and explain the respective protection mechanisms. Note that the attack vectors discussed in Chapter 4 are basic tools for $\mathcal{A}$ to influence RS's data, as the directly manipulated trust views are finally uploaded to RS.

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

Whenever $\mathcal{A}$ is able to register itself with several forged identities at RS (or $\mathcal{A}$ can control the systems of already registered entities), $\mathcal{A}$ can upload manipulated trust views in their name to RS. As RS generates its opinions based on the trust views of its users, $\mathcal{A}$ 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 $\mathcal{A}$ controls a large enough fraction of RS's user base, $\mathcal{A}$ effectively controls the content of the recommendations generated by RS.

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

Furthermore, the selection strategy for the computation of recommendations using similarity weighting (cf. Section 5.2.2) provides protection against Sybil attacks (cf. [68] for a similar approach). When RS is requested to calculate a recommendation on the issuer trust for a CA, only those trust views are selected that are similar to the one of the requesting entity $\mathcal{E}_1$. However, $\mathcal{E}_1$'s trust view in general is unknown to $\mathcal{A}$. A trust view uploaded by $\mathcal{A}$ is only by chance considered during the calculation. This makes it difficult for $\mathcal{A}$ to generate and submit trust views to manipulate the recommendation without knowing the $\mathcal{E}_1$'s trust view. The trust view is only communicated over secured connections. Thus, gathering information about trust views requires observing the TLS traffic of the $\mathcal{E}_1$ or sophisticated social engineering attacks. Even if $\mathcal{A}$ manages to tailor the manipulated trust views for one target entity, the overall success is limited.

Moreover, proactive techniques can be implemented to protect RS against Sybil attacks. The upload of a large amount of trust views within a certain time interval or sudden changes of a CA's issuer trust within trust views can be statistically detected as shown in [73]. Besides that, Sybil attacks can be lessened when RS considers only trust views or trust assessments of a certain minimal age. The presented techniques significantly increase the costs of an attack, as they increase the time span during which the attack is to be executed. This helps to bridge the gray period. After the fraudulent certificate has been revoked or blacklisted, the attack is anyway without effect.

Attacker goals and defenses

The previous subsection discussed the attack vector of Sybil attacks that $\mathcal{A}$ can use to manipulate the recommendations provided by RS. In this section, specific attacker goals, their possible realization, and how they apply to CA-TMS are discussed.

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

A self-promoting attacker can approach his goal by a Sybil attack on RS. To use a Sybil attack, $\mathcal{A}$ must overcome the above described defense mechanisms. Additionally, as RS is only queried when a trust assessment is newly initialized, the CA controlled by $\mathcal{A}$ must be new to $\mathcal{E}_1$. On the other hand, due to the use of validation services, the CA must be legitimately observed first, which in fact can hardly be steered by $\mathcal{A}$ for multiple entities. Even for a single entity, this requires social engineering coordinated with the Sybil attack and perfect timing.

Slandering In opposition to a self-promoting attack, slandering aims at lowering the reputation of a specific CA. As $\mathcal{A}$ does not directly benefit from decreased trust in CAs, he might only aim at disturbing the proper functioning of CA-TMS.

The only possibility is to use a Sybil attack on RS. Again, $\mathcal{A}$ must overcome the defense mechanisms and may only influence newly initialized trust assessments on $\mathcal{E}_1$'s side. During the bootstrapping of a trust view, this has a certain impact, but afterward, the attack is of limited relevance. Furthermore, the attack in the worst case increases the number of required reconfirmations. In this case, local experiences are collected, which leads to a fade out of the influence of the manipulated recommendation.

Whitewashing Whitewashing describes the approach of an entity with negative reputation to re-appear under a new, clean identity. Whitewashing was already shown not to apply to the Web PKI in Chapter 4.

5.4.3 Reputation system performance

The performance evaluation is based on simulations using 64 real browsing histories also used in Section 4.4. To evaluate the effects of the reputation system, the evolution of each trust view is simulated with and without the use of the reputation

system and the resulting trust views are compared. For the simulation, the 63 trust views derived from the remaining browsing histories serve as the knowledge base from which the reputation system aggregates the recommendations.

The remainder of this section is organized as follows. First, we show the suitability of the Jaccard similarity index, which is a central tool to the reputation system for trust view pre-selection and weighting during the computation of recommendations. Then, the improvements of information collection provided by the reputation system are presented. Third, the influence of the reputation system on the attack surface is evaluated. The last part of this section is focused on data loads and communication overheads.

Suitability of the Jaccard similarity index

In the following, we show that the Jaccard similarity index is a suitable measure to facilitate opinion aggregation.

Suitability for pre-selection We first show that the Jaccard similarity index is suitable for pre-selecting trust views as potential input for the aggregation of a recommendation. To test this, the probability that a trust assessment for a requested CA is contained within the pre-selected trust views is evaluated depending on the Jaccard similarity.

The tests are as follows: One trust view View_i serves as the one of the requester and one of the trust assessments in View_i serves as the requested trust assessment TA. TA is removed from View_i and the Jaccard similarity $J(\mathsf{View}_i^*, \mathsf{View}_j)$ of the modified trust view View_i^* to all other trust views $\mathsf{View}_j,\ j \neq i$ is computed. Now, for $x \in \{0, 1, \ldots, 19\}$ let the intervals $\mathcal{I}_x$ be defined as $\mathcal{I}_x =]a_x, b_x]$ with $a_x = x \cdot 0.05$ and $b_x = (x+1) \cdot 0.05$. Further let $M_{\mathcal{I}_x,\mathsf{TA}} = \{\mathsf{View}_j | J(\mathsf{View}_i^*, \mathsf{View}_j) \in \mathcal{I}_x \wedge \mathsf{TA} \in \mathsf{View}_j\}$ and $M_{\mathcal{I}_x} = \{\mathsf{View}_j | J(\mathsf{View}_i^*, \mathsf{View}_j) \in \mathcal{I}_x\}$. Then, the probability to find TA in the considered trust views is computed as:

$$pr(\mathsf{TA}) = \frac{|M_{\mathcal{I}_x,\mathsf{TA}}|}{|M_{\mathcal{I}_x}|}.$$

The test is repeated for each trust assessment in View_i as well as for each possible choice of View_i. The computed probabilities are averaged.

The results of this test are shown in Figure 5.3. The probability to find TA in a pre-selected trust view grows with growing Jaccard similarity. From the results it can be deduced that pre-selecting trust views based on the Jaccard similarity and subsequently not considering trust views with low similarity does not lead to a loss of information.

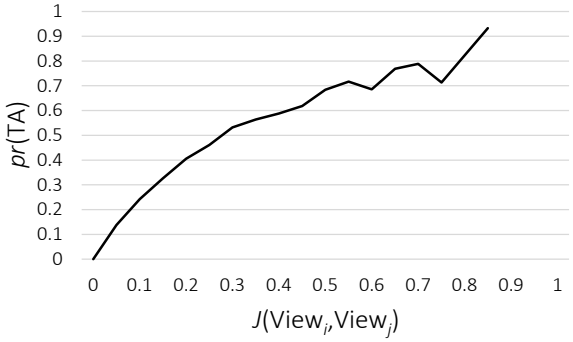
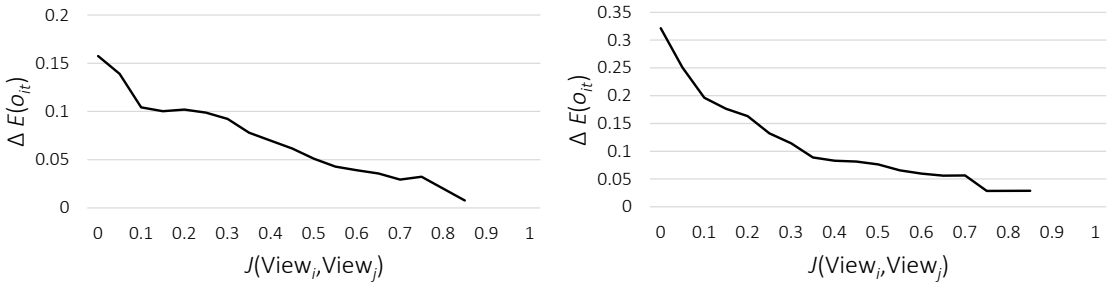


Figure 5.3: Probability $pr(\mathsf{TA})$ to find a trust assessment for a newly observed CA in another trust view depending on the Jaccard similarity.

Suitability for similarity weighting We show that the Jaccard similarity is suitable to group trust views according to user groups within which the trust values are comparable. This is directly related to the recommendation quality, when using weights $w_i = J(\mathsf{View}, \mathsf{View}_i)$ to aggregate a recommendation as defined in Section 5.2.2.

To test this, the deviation of the expectations depending on the Jaccard similarity of the trust views is measured. We first compute the similarity between each pair of trust views $J(\mathsf{View}_i, \mathsf{View}_j),\ i, j \in \{1, ..., 64\},\ i \neq j$ and group them pairwise into $G_{\mathcal{I}_x} = \{(\mathsf{View}_i, \mathsf{View}_j) | J(\mathsf{View}_i, \mathsf{View}_j) \in \mathcal{I}_x\}$ with $\mathcal{I}_x$ as defined above. Then, within each $G_{\mathcal{I}_x}$ the average deviation of the expectation values for the issuer trusts $\Delta E(o_{it}^{ca})$ and $\Delta E(o_{it}^{ee})$ over all pairwise joint trust assessments is computed.



(a) **(b)**

Figure 5.4: Average deviation of (a) $E(o_{it}^{ca})$ / (b) $E(o_{it}^{ee})$ depending on the Jaccard similarity.

The results are shown in Figures 5.4a and 5.4b. With growing Jaccard similarity the deviation of the expectation of opinions concerning common trust assessments shrinks. That means trust views with similar CA sets on average also contain similar opinions on the common CAs. Thus, the Jaccard similarity index is suitable

for similarity weighting during the aggregation of a recommendation. It puts the more weight on opinions, the better they reflect the requesting entities final opinion.

From these results, we conclude that the Jaccard similarity is suitable to preselect the relevant trust views for the aggregation of recommendations and provides a natural weighting for the influence of single opinions into the aggregated recommendation.

Improvement of information collection

We show that the use of the reputation system significantly improves the information collection of CA-TMS and thus speeds up the bootstrapping of an entity's trust view. This is shown based on reconfirmation rates. The need for reconfirmations represents the absence of sufficient information for decision making. Thus, lower rates imply that more information is available to the system at a certain point in time, while achieving the same rate at an earlier point in time shows the speed up in information collection.

We measure the average reconfirmation rate with and without the use of the reputation system. For each simulation run, we select one of the 64 browsing histories and simulate the evolution of the trust view. The 63 trust views derived from the remaining 63 histories form the knowledge base of the reputation system. The reconfirmation rates for the three different security levels $l_{max} = 0.95$, $l_{med} = 0.8$ and $l_{min} = 0.6$ are measured independently. The simulation results are averaged over all simulation runs.

Figure 5.5 shows the improvements achieved with the reputation system concerning reconfirmation rates. It shows how the percentage of hosts, for which a reconfirmation is required, develops on average over the course of the evolution of

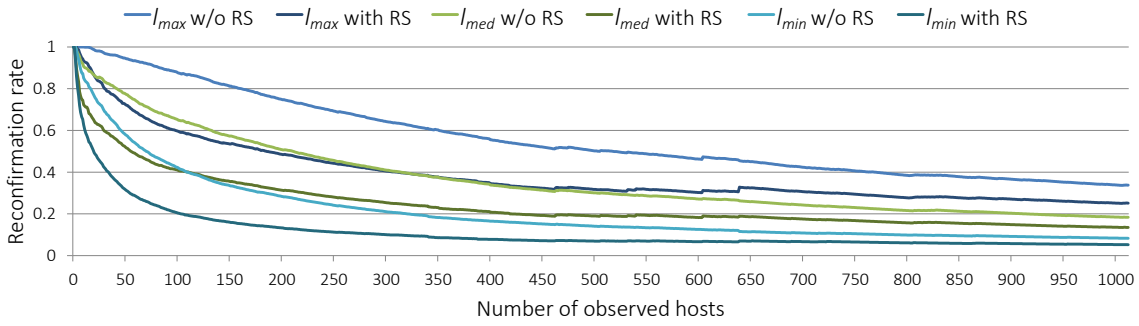


Figure 5.5: Percentage of observed hosts for which a reconfirmation was required with and without the use of the reputation system, concerning the security levels l_{max}, l_{med} and l_{min}.

trust views. This is shown for the three different security levels.

For each of the different security levels, the reconfirmation rate with the use of the reputation system lies significantly below the rate without reputation system. For example, given a required security level of l_{min}, the use of the reputation system reduces the reconfirmation rates for the first one hundred hosts from 42% to 20%. The differences of the respective rate with and without the reputation system after a given number of observed hosts are depicted in Figure 5.6. Note that several histories in our data set end at around 630 hosts leading to the gap in the averaged rates. The differences are higher in the early phase of the trust view evolution because the rates themselves also drop with a growing number of observed hosts. This means that the reputation system has the strongest effects at the beginning of the trust view evolution where an accumulation of reconfirmations occurs and thus is most important for the user experience. In summary, the reconfirmation rates drop faster and stay below the rates without reputation system. Hence, a user of CA-TMS is confronted with less delays due to reconfirmations during the bootstrapping phase.

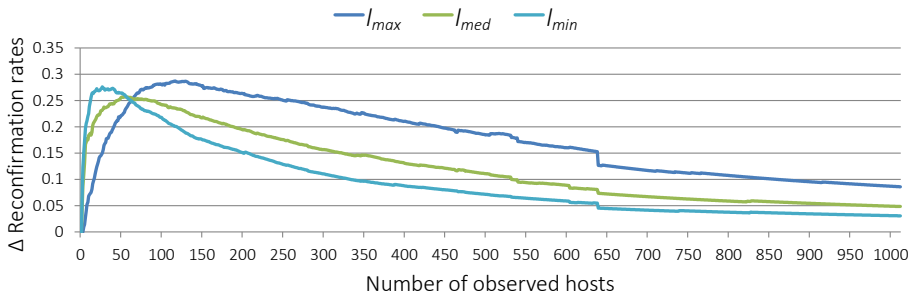


Figure 5.6: Differences in reconfirmation rates for security levels l_{max}, l_{med} and l_{min}.

A second measure that shows the bootstrapping speed-up by the reputation system is the difference of hosts that need to be accessed until a comparable level of the reconfirmation rate is reached with and without the reputation system. As a reconfirmation represents the lack of information for decision making, reconfirmation rates indirectly measure the amount of collected CA trust information. Figure 5.7 depicts the differences of required hosts until certain target reconfirmation rates are achieved. The differences are given as percentage of the hosts that need to be accessed when no reputation system is used. Figure 5.7 shows that the numbers of required hosts for experience collection are 50% - 85% lower due to the reputation system. This shows the speed-up of the bootstrapping due to the reputation system.

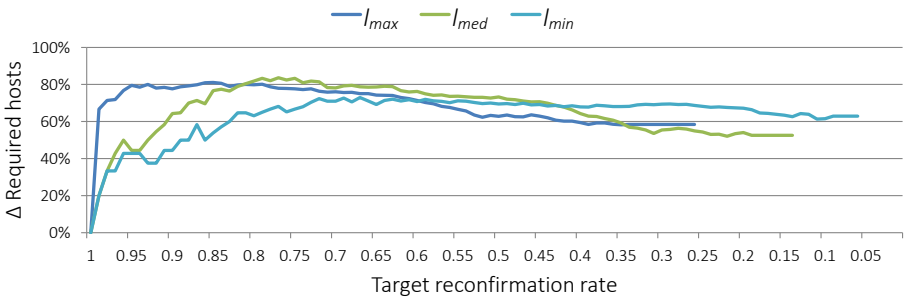


Figure 5.7: Differences in required hosts to reach target reconfirmation rates for security levels l_{max}, l_{med} and l_{min}.

Effects on the attack surface

In this section it is shown, how the reputation system affects the individual size of the attack surface. Again this is based on simulations based on the 64 browsing histories. Recall that in the simulations no negative experiences are included, meaning the results lead to an upper bound for the attack surface.

Figure 5.8 shows the measurements for the average difference in the expectation of issuer trust opinions assigned to the CAs between simulation runs with and without the reputation system. The deviations have their maximum for trust views of low maturity, namely between 10 and 30 observed hosts. Afterwards, the deviations fade the more experiences are collected. This shows that the reputation system does not simply lead to an increase of the trust values, but boosts them towards their final state. Besides that, the deviation is always smaller than 0.2. This means that the trustworthiness of a CA is at most increased to the next security level by the reputation system.

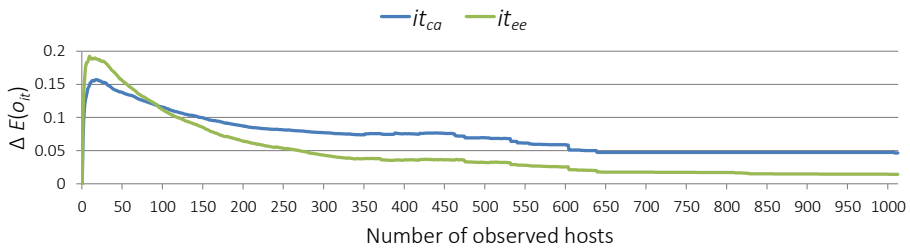


Figure 5.8: Average difference in expectation values $E(o_{it}^{ca})$ and $E(o_{it}^{ee})$

Figure 5.9 shows the average number of CAs trusted for the different security levels with and without the reputation system. These numbers directly correspond to the size of the attack surface (according to the metric presented in Section 4.4.2) for the three extreme cases where either l_{max}, l_{med} or l_{min} is assigned to all domains.

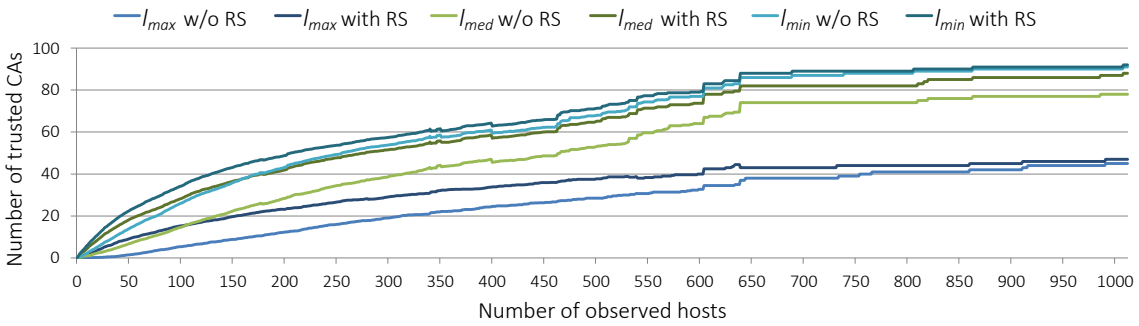


Figure 5.9: Number of CAs trusted to issue certificates concerning the required security levels l_{max}, l_{med} and l_{min} with and without reputation system.

Figure 5.9 shows that the speed-up achieved by the reputation system comes at the cost of a slight increase of the attack surface compared to CA-TMS without reputation system. This results from the additional trust information which allows that some CAs reach the next security level at a lower number of observed hosts. The highest difference is observable for the security level l_{med}, while the effect on the attack surface concerning the security levels l_{max} and l_{min} are much smaller. This shows, that mainly CAs are shifted from the class of minimum trustworthy CAs to the class of medium trustworthy CAs. The differences in the sizes of the attack surfaces fade, the more experiences are collected, i.e. the more evolved the trust views are. This again shows, that the reputation system evolves the trust views towards their final state.

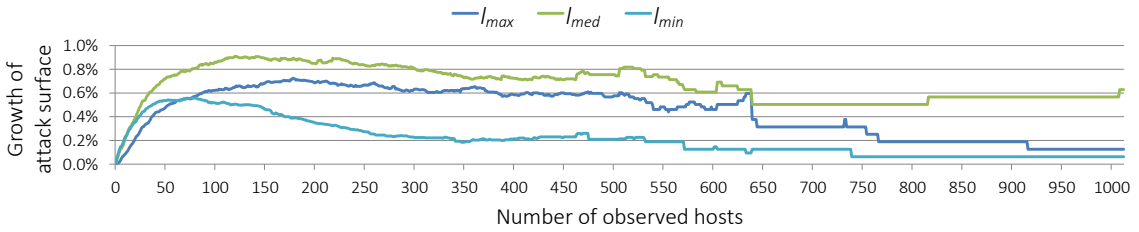


Figure 5.10: Growth of the attack surface due to the reputation system concerning the required security levels l_{max}, l_{med} and l_{min} in relation to the attack surface spanned by the 1,590 trusted CAs of the Web PKI.

Figure 5.10 depicts the effects of the reputation system on the attack surface in relation to the system-centric setting with 1,590 trusted CAs, where the attack surface is defined as $AS = \sum_{\texttt{CA} \in \mathsf{PKI}} dom\,[\texttt{CA}]$. $dom\,[\texttt{CA}]$ is the number of domains which CA is allowed to sign. PKI describes the set of all CAs which are part of the Web PKI. Figure 5.10 shows that the set of CAs additionally trusted due to the use

of the reputation system makes up for less than 1% of the attack surface AS. The maximum attack surface increase again is reached for the security level l_{med}.

In summary, the reputation system provides a speed-up of the bootstrapping of more that 50% while leading to minimal increases of the attack surface of less than 1%.

Data loads and communication overheads

The use of the reputation system reduces the number of reconfirmations. However, the reputation system is queried for recommendations. We compare the traffic overheads and data loads in the two settings during browsing. We assume, that the user is registered at the reputation system and its current trust view is already available to the reputation system. Thus, requesting a recommendation for a new CA requires the transmission of the CAs name and its public key, which can be realized by sending the CA's certificate to the reputation system. A reconfirmation request to a validation service also requires the transmission of the certificate in question. The exact message sizes depend on the respective implementation, however wlog. it can be assumed that the message sizes of the requests are of comparable size. The same holds for the size of the response messages. Thus, the evaluation focuses on the number of requests and their timing criticality.

The number of reconfirmations is counted during the simulation runs, while the number of queries to the reputation system is given by the number of CAs. We present the differences depending on the number of observed hosts in Table 5.1. It shows that the absolute number of queries to the reputation system is higher than the number of saved queries to validation services, which yields additional traffic. This has several reasons. First, newly observed CAs are always reconfirmed whether or not the reputation system recommends the CA as trustworthy to guarantee that adding a new CA to the trust view is always justified. Furthermore, despite a recommendation the derived trust level might not be high enough for the acceptance of a certificate.

However, there is a fundamental difference between requests to validation services and requests to the reputation system. The first are blocking, i.e., browser page loading must be delayed until the response of the validation services is obtained, while requests to the reputation system provide information for future browsing and can be done in the background. These requests can even be delayed if the reputation system is temporarily not available as they are transparent to the user.

The overall traffic considering requests to the reputation system per time interval depends on the number of users and the number of new CAs they observe on average.

		Number of observed hosts					
		25	50	100	200	400	600
l_{max}	req(VS)	25	47	88	150	224	278
	req_{RS}(VS)	21	36	60	97	140	182
	Δ req(VS)	4	11	28	53	84	96
	req(RS)	27	41	57	79	99	122
l_{med}	req(VS)	21	39	65	102	137	163
	req_{RS}(VS)	16	26	41	63	84	110
	Δ req(VS)	5	13	24	39	53	53
	req(RS)	27	41	57	79	99	122
l_{min}	req(VS)	18	29	42	57	66	75
	req_{RS}(VS)	12	16	21	27	31	40
	Δ req(VS)	6	13	21	30	35	35
	req(RS)	27	41	57	79	99	122

Table 5.1: Numbers of requests to validation services and the reputation system for the required security levels l_{max}, l_{med} and l_{min}. req(VS) denotes the number of requests to validation services in the setting without reputation system, req_{RS}(VS) the respective number when the reputation system is used. Δ req(VS) denotes how many requests to validation services are saved due to the use of the reputation system and req(RS) denotes the number of requests to the reputation system.

In our data sets we found an average of seven new CAs per month. Thus, the reputation system is on average queried seven times a month per registered user. For example, assuming a user base of one million users would yield 162 requests per minute. Peaks that might come up, e.g., because of day and night rhythms, can easily be balanced because the requests to the reputation system are not time critical.

5.4.4 Push service evaluation

In this section the proposed push service is evaluated in terms of success probabilities of an attacker. We compare the system-centric setting with the user-centric setting without and with the push service and show the security gain by such a service. First, the detection capability for behavioral changes of our system depending on the class of the compromised CA is evaluated. Based on these findings, the success probabilities of an attacker are presented.

Detection of behavioral changes

The performance of the detection mechanism is evaluated in terms of successfully attacked entities until the service provider SP detects the maliciously issued certificate due to reports send by its clients. For the analysis we consider an attacker $\mathcal{A}$ according to the attacker model given in Section 3.1.2. $\mathcal{A}$ possesses a fraudulent certificate C for the web service $\mathcal{S}$ issued by CA. $\mathcal{A}$ uses C to attack relying entities when connecting to $\mathcal{S}$. Thus, we consider the users of $\mathcal{S}$ as the target group G of attacked relying entities. $\mathcal{A}$ could also attack several web services in parallel, yet this would only increase the size of the target group. Thus, for the analysis we assume, that $\mathcal{A}$ only attacks one web service at a time.

Let q be the percentage of entities G that consider CA as trustworthy to issue certificates for $\mathcal{S}$ in the user-centric setting. Trusting CA to issue certificates for $\mathcal{S}$ means that CA reaches a sufficiently high issuer trust for end entity certificates in the respective trust views. We call q the trust rate. Then, the probability that for entity $\mathcal{E}^*$, randomly chosen from G, trust validation outputs `trusted` for C is q and the probability that it outputs `untrusted` and $\mathcal{E}^*$ subsequently reports the attack to SP is $(1-q)$. Thus, the probability that the n-th entity detects the fraudulent certificate when connecting to $\mathcal{S}$ is $pr_{det}(n) = q^{n-1} \cdot (1-q)$. The expectation value for the number of successfully attacked connections to $\mathcal{S}$ until a detection occurs, can be computed as:

$$Exp_{con} = \sum_{n=1}^{\infty} pr_{det}(n) \cdot n = \sum_{n=1}^{\infty} q^{n-1} \cdot (1-q) \cdot n = \frac{1}{1-q} \qquad [2]$$

From this follows, that given q per cent of the entities trust CA, $\mathcal{A}$ on average can successfully attack $Exp_{con} - 1 = \frac{1}{1-q} - 1$ entities until the attack is detected and reported to SP. Table 5.2 shows the average number of successfully attacked entities for different values of q. The numbers show that an attacker must compromise a CA that achieves a trust rate approaching one within the target group in order not to be rapidly detected. Even for $q = 0.99$ an attack is on average detected after 99 successfully attacked entities.

q	0.1	0.3	0.5	0.7	0.9	0.95	0.99
$Exp_{con} - 1$	0.111	0.429	1	2.333	9	19	99

Table 5.2: Number of on average successfully attacked connections depending on q

[2] $\sum_{n=1}^{\infty} q^{n-1} \cdot (1-q) \cdot n = \sum_{n=1}^{\infty} q^{n-1} \cdot n - \sum_{n=1}^{\infty} q^n \cdot n = \sum_{n=0}^{\infty} q^n \cdot (n+1) - \sum_{n=1}^{\infty} q^n \cdot n = \sum_{n=0}^{\infty} q^n \cdot n + \sum_{n=0}^{\infty} q^n - \sum_{n=1}^{\infty} q^n \cdot n = q^0 \cdot 0 + \sum_{n=1}^{\infty} q^n \cdot n + \sum_{n=0}^{\infty} q^n - \sum_{n=1}^{\infty} q^n \cdot n = \sum_{n=0}^{\infty} q^n = \frac{1}{1-q}$

p	l_{min}	l_{med}	l_{max}
≥ 0.1	82 (26.8%)	63 (20.6%)	30 (9.8%)
≥ 0.2	57 (18.6%)	42 (13.7%)	24 (7.8%)
≥ 0.3	40 (13.1%)	29 (9.5%)	15 (4.9%)
≥ 0.4	30 (9.8%)	16 (5.2%)	9 (2.9%)
≥ 0.5	19 (6.2%)	13 (4.2%)	8 (2.6%)
≥ 0.6	16 (5.2%)	11 (3.6%)	5 (1.6%)
≥ 0.7	12 (3.9%)	7 (2.3%)	2 (0.7%)
≥ 0.8	7 (2.3%)	4 (1.3%)	0 (0.0%)
≥ 0.9	0 (0.0%)	0 (0.0%)	0 (0.0%)

Table 5.3: Number of CAs (per cent of total) for which at least the trust rate q is reached for different required security levels for the attacked web service. The percentage refers to the CA super set of 306 CAs found together in the analyzed 64 trust views.

Table 5.3 shows how many CAs achieve a certain trust rate q in our data set of 64 trust views. The super set of CAs found together in the 64 trust views contains 306 different CAs. For example, it can be observed that if entities require a security level l_{min} for $\mathcal{S}$, then only 26.8% of the CAs in the super set reach a trust rate of $q \geq 0.1$. For l_{max} this even shrinks to 9.8%. None of the CAs reaches a trust rate of $q = 0.9$. Additionally, all CAs not contained in the super set are not trusted by any entity in the target group. Thus, a fraudulent certificate issued by one of those CAs would be immediately detected. In summary this shows, that in practice, it is very improbable that $\mathcal{A}$ can identify and compromise a CA that achieves a high trust rate in the target group, which would prevent an immediate detection of the attack. Even compromising the CA which issued the legitimate certificates for the $\mathcal{S}$ is no guarantee for a high trust rate, because the trust views of the attacked entities might lack additional positive experiences for this CA for other web services.

Attacker success probabilities

The success probability of $\mathcal{A}$ is evaluated based on the trust rate q of the CA from which $\mathcal{A}$ obtained the fraudulent certificate. In the previous section it was shown, after how many attacked connections the attack is detected on average. With the use of the push service, $\mathcal{A}$'s success probability depends on the number of attacked connections and drops to zero once the attack is detected. Most important, this bounds the absolute number of entities that can be successfully attacked independent from the size of the target group.

In the user-centric setting without a push service, the success probability directly

depends on q, while the success probability in the system-centric setting is 100%. This is depicted in Table 5.4. The evaluation concerns the time period between the issuance of the fraudulent certificate and its revocation, from which on the certificate cannot be used for an attack in any setting. However, while a revocation of the fraudulent certificate protects from the misuse of this single certificate, a warning by the push service leads to suspending the CA, which means further fraudulent certificates issued by the same CA will also be detected.

Setting:	system-centric	user-centric	user-centric with push service
Success probab.	100%	q	q^n

Table 5.4: Success probability for an attack depending on q reached by the CA that issued the fraudulent certificate. n denotes the number of entities $\mathcal{A}$ has attacked prior to this attack.

5.5 Conclusion

To conclude this chapter we summarize the results. Service providers for CA-TMS realizing a reputation system and a push service for CA warnings were presented. The services have been evaluated in terms of security and performance. It was shown that the reputation system speeds up the information collection and thus the bootstrapping by more than 50% while only minimally increasing the attack surface by less than 1% compared to the strictly local information collection. Differences in trust views resulting from the use of the reputation system fade out, the more evolved a trust view becomes. This shows, that the reputation system fastens the evolution of trust views while not changing their individual character. Traffic overheads are kept at a low rate of approximately seven requests per month on average per user of the reputation system. These requests are non-blocking and thus do not lead to any delays during browsing.

The presented push service solves the problem of relying entities making decisions based on outdated information in case a CA changes its behavior from trustworthy to untrustworthy. It was shown that the presented detection mechanism is capable to detect maliciously behaving CAs, even if a majority of relying entities already trusts the CA in question. Moreover, the push service puts an absolute bound on the number of entities that can be attacked until the attack is detected. A detection

leads to an immediate warning of the relying entities and the elimination of the threat. For our test group of Internet users, the analysis showed that there does not exist a single CA which is trusted by at least 90% of the group members. This illustrates the detection and attack prevention capability in practice. Even a trust rate of 90% in the target group of the attack on average leads to a detection after 9 attacked entities.

In summary, the proposed service providers highly improve the user experience of CA-TMS because of the speed-up of information collection and thus the prevention of delays and the reduction of the dependence on external validation services. At the same time, service providers impose strong security benefits by adding highly efficient detection mechanisms for fraudulent certificates and the subsequent prevention of attacks.

6 CA revocation tolerant PKI

In this chapter, we present a solution for the too-big-to-fail problem of CAs. This problem refers to the impossibility to revoke the key (and the according certificate) of a large CA, even if actually required from a security perspective. In public key infrastructures, revocation of a certificate is required whenever a key needs to be invalidated before the end of its validity period. For example, in order to prevent misuse when a key was compromised. However, the revocation of a CA's key subsequently invalidates all certificates that contain the revoked key in their certification path. This in turn means, that all services using such an invalidated certificate for authentication become unavailable until their certificates are exchanged by new ones. In many scenarios, such a temporal unavailability of services is not acceptable and thus, either requires that the revocation of a CA key is considerably delayed or even completely avoided. In Chapter 3, it was shown that this behavior is a huge problem in practice and leads to security flaws.

In Section 6.1, we present a solution to the too-big-to-fail problem of CAs. The problem is solved with forward secure signature schemes (FSS). FSS provide a (partial) chronological ordering of the signatures generated with one key. It is guaranteed, that even an adversary that compromises the key cannot manipulate the ordering of previously generated signatures. This property allows to revoke a key without invalidating former signatures, thus the unavailability of dependent services is prevented. We provide the concepts and implementation details concerning changes in the standard path validation and revocation mechanisms and protocols. We show how to implement the solution in the Web PKI in Section 6.2.

In Section 6.3, the solution is evaluated in regard to practicality. For the evaluation we use a reference implementation of XMSS [12], a hash-based FSS. It is shown that our solution provides good performance and practical data loads by presenting runtimes as well as certificate and signature sizes. These are compared to the standard setup where common signature schemes like RSA or (EC) DSA are used. Furthermore, we compare our solution to an alternate approach based on time-stamps and show that time-stamps are not a viable solution to the too-big-to-fail

problem. Section 6.4 concludes this chapter.

The contributions of this chapter were published as parts of [B10]. The publication covers the theoretical results presented in Section 6.1. This chapter extends these results with implementation details and an evaluation regarding the practicality of the solution.

6.1 Realizing CA revocation tolerance with forward secure signatures

In Chapter 3 it was shown that the too-big-to-fail problem results from the implicit revocation of all certificates that rely on a revoked CA certificate, i.e. that have the revoked CA certificate in their certification path. To resolve this problem, revocation of CA certificates and the according revocation checking must be realized such that legitimately issued certificates stay valid despite a revocation of superordinate CA certificates. This requires the possibility to securely distinguish between legitimate signatures and such generated by an attacker who compromised a CA's private signing key. We call a PKI that realizes the property of preserving the validity of legitimately issued certificates in the face of CA certificate revocations a *CA revocation tolerant PKI*.

In the following, a CA revocation tolerant PKI is achieved by replacing the conventional signature schemes used for certificate signing by forward secure signature schemes (cf. Section 2.1.2) in combination with an adapted version of the chain model for path validation. End entities further use conventional signature schemes, such as RSA and (EC) DSA, to minimize the deployment efforts. For authentication purposes, one does not gain a benefit from using FSS because signatures are in general verified in close temporal proximity to signature creation. An extension of the CA revocation tolerant PKI to FSS also being used by end entities in scenarios where non-repudiation is required is presented in Chapter 7.

An exemplary certification path given a CA revocation tolerant PKI realized with an FSS is shown in Figure 6.1. In each certificate an FSS public key is certified except for the end entity certificate. The signatures on all certificates are generated using the FSS.

In the following, we present the solution in detail. First, it is explained how FSS enable to securely distinguish between legitimate signatures and such generated by an attacker. Then, fine grained revocation is presented, which makes use of the FSS' special properties during the revocation of a certificate. Finally, we present the adapted version of the chain model for path validation.

6.1.1 The chronological ordering of signatures

Recall that in FSS, as presented in Section 2.1.2, each signature is accompanied by an index specifying the interval of the key pair's lifetime in which the signature was generated. By this index, the signatures are chronologically ordered. Concerning signatures generated prior to a compromise, this ordering is immutable. Manipulating the ordering would require the generation of signatures for past intervals. This is prevented by the forward security property. Thus, given the index of the interval in which a key compromise occurred, distinguishing between legitimate and fraudulent signatures can be done by comparing indices. How to identify the index of the interval in which the key compromise occurred depends on the way how the intervals are defined for the used FSS. This is discussed in Section 6.2.2.

If the used FSS allows multiple signatures to be generated in one interval, the ordering is only a partial ordering. For the signatures generated in the same interval, fraudulent and legitimate signatures cannot be distinguished based on indices. Therefore, FSS that evolve the key after each signature generation are optimal regarding this property. However, even for FSS that allow multiple signatures in one interval, the number of signatures for which the origin is unclear is limited.

6.1.2 Fine grained revocation

The forward security property allows to handle revocation in a fine grained manner. In case of a key compromise, the forward security property guarantees that all signatures created prior to the compromise originate from the certificate owner. So there is no need to render these signatures invalid. As all signatures contain the index of the interval they were created in, the validity of the certificate is not revoked in general, but only for all intervals starting from the interval when the key was compromised. Given the compromise happened in the interval with index c, the revocation starts at index c. A signature including index i is accepted as valid if $i < c$

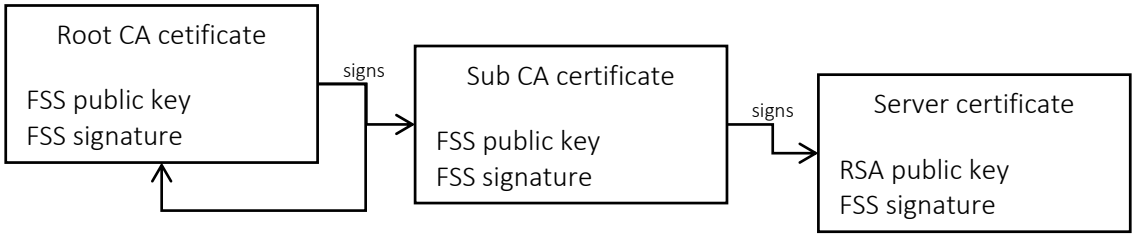


Figure 6.1: Examplary certification path in an CA revocation tolerant PKI with one intermediate Sub CA

and invalid if $i \geq c$. Thus, a revocation of a certificate additionally has to include a revocation index. How this can be realized with current revocation mechanisms is shown in Section 6.2.2. For FSS which allow multiple signatures in one interval, fine grained revocation invalidates all signatures whose origin is unclear. This may also affect a limited number of legitimate signatures but is required to maintain security.

6.1.3 Adaptation of the validity model

The chronological ordering of signatures and fine grained revocation must be considered during path validation to realize a CA revocation tolerant PKI. An appropriate validity model is given in Definition 6.1, which we call *chain model for FFS with time limitation*. It is an adapted version of the original chain model combined with properties of the shell model. Both were given in Definitions 2.1 and 2.3 in Section 2.2.4.

For the definition let $n \in \mathbb{N}$ be the length of the certification path $p = (C_1, \ldots, C_n)$. C_1 is the self-signed certificate of the Root CA. C_n is the certificate of the end entity. We denote by $T_i(k)$ the starting date of the validity period of C_k and by $T_e(k)$ its expiration date. Additionally, let $I_s(k)$ be the signing index used to sign certificate C_k and $I_r(k)$ a possible revocation index for certificate C_k. Further let $I_r(k) = \infty$ in case there is no revocation for certificate C_k. Note that $I_s(k)$ and $I_r(k-1)$ are indices belonging to the same key pair. The revocation of end entity certificates is done in the conventional way.

Definition 6.1 (Chain model for FFS with time limitation)**.** *Given all signature schemes involved in the certification path are FSS except for the end entity scheme, then a certification path is valid at verification time T_v if:*

1. *C_n is valid at verification time T_v: $T_i(n) \leq T_v \leq T_e(n)$ and C_n is not revoked at T_v.*

2. *Every CA certificate in the path is valid at the verification time T_v: $T_i(k) \leq T_v \leq T_e(k)$ and not revoked for the signing index $I_s(k+1)$ used for the subordinate certificate in this path: $I_s(k+1) < I_r(k)$ for all $1 \leq k \leq n-1$.*

The adapted validity model is required, because the shell model, which is the standard validity model in the Web PKI [83], does not allow the consideration of a chronological ordering of signatures. The validity of all certificates in the certification path is evaluated at the time of signature validation and no individual differentiation is made for the signatures that are verified along the certification path. In the chain model, this is addressed based on the signature generation times.

The chain model considers a certificate as valid if it is not revoked nor expired and was issued before a potential superordinate CA certificate revocation or expiration. As a consequence, subordinate certificates remain valid, even after the invalidation of superordinate certificates. However, the original chain model (cf. Definition 2.3) has a drawback. It allows signature validation at any point in the future. This implies that the according revocation information for each involved certificate must be stored for an indefinite time, because the time when a potential attacker makes use of a compromised key cannot be controlled. However, in the Web PKI the availability of revocation information is in general only guaranteed until the expiry of the according certificate [81]. The chain model for FFS with time limitation resolves these issues. Certification paths are invalidated upon the expiry of one of the included certificates, while revocations are handled in the fine grained manner as described above.

6.2 Implementation in the Web PKI

In this section, it is shown how the Web PKI can be transitioned to a CA revocation tolerant PKI. We note that a standardization of the FSS itself is required to allow interoperability. This standardization is not part of this thesis. However, we highlight where standardization is required.

First the representation of FSS keys within X.509 certificates is explained. Then, the adaptation of revocation mechanisms in order to cover fine grained revocation is presented. Finally, we discuss necessary changes in path validation specified in RFC 5280 [83] and give an overview over deployment efforts.

6.2.1 FSS and X.509 certificates

At first we consider the use of FSS in X.509 certificates. It is shown how keys are represented and how this can be mapped to the requirements of FSS. X.509 certificates are specified in the abstract syntax notation version 1 (ASN.1)[96]. Listing 6.1 shows the ASN.1 representation of an X.509 certificate. There are three fields, where signature algorithm and key information are stored in a certificate: the `signatureAlgorithm` field and the `signature` and `subjectPublicKeyInfo` fields within the `tbsCertificate` element. The first two fields are of type `Algorithm-Identifier` and contain the identifier for the cryptographic algorithm used by the CA to sign this certificate. Signature algorithms that are supported are listed in [80, 105, 99]. However, it is explicitly allowed to support additional signature algorithms.

```
Certificate  ::=  SEQUENCE  {
        tbsCertificate        TBSCertificate,
        signatureAlgorithm    AlgorithmIdentifier,
        signatureValue        BIT STRING  }

 TBSCertificate  ::=  SEQUENCE  {
        version          [0]  EXPLICIT Version DEFAULT v1,
        serialNumber          CertificateSerialNumber,
        signature             AlgorithmIdentifier,
        issuer                Name,
        validity              Validity,
        subject               Name,
        subjectPublicKeyInfo  SubjectPublicKeyInfo,
        issuerUniqueID   [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                              -- If present, version MUST be v2 or v3
        subjectUniqueID  [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                              -- If present, version MUST be v2 or v3
        extensions       [3]  EXPLICIT Extensions OPTIONAL
                              -- If present, version MUST be v3  }

   SubjectPublicKeyInfo  ::=  SEQUENCE  {
        algorithm             AlgorithmIdentifier,
        subjectPublicKey      BIT STRING  }

   AlgorithmIdentifier  ::=  SEQUENCE  {
        algorithm               OBJECT IDENTIFIER,
        parameters              ANY DEFINED BY algorithm OPTIONAL  }
```

Listing 6.1: X.509 certificate [83]

The `subjectPublicKeyInfo` field contains the public key, that is certified with this certificate and additionally the algorithm to be used with the certified key. The algorithm in the `subjectPublicKeyInfo` is as well of type `AlgorithmIdentifier`. Thus, whether the CA's signing key or the certified key is to be used with a FSS, this is specified in the same way.

The algorithm identifier is used to identify a cryptographic algorithm using an OID and optionally allows to specify parameters. These parameters may vary depending on the algorithm.

There are two possibilities to specify parameters for an algorithm. Either they are encoded within the OID of the algorithm as proposed in [101], meaning there is a standard set of parameters for the given algorithm. Or, they are explicitly stated within the `parameters` field of an `AlgorithmIdentifier` [83].

Thus, the use of FSS is covered by the X.509 standard and does not require any changes. The FSS algorithm itself needs to be standardized and have an OID assigned. This OID needs to directly identify the required parameters or, this standardization must include a definition of required parameters to be included in respective `AlgorithmIdentifier` fields in the certificate. For examples of such parameter

sets please refer to [80]. Concerning FSS, the parameters might in particular include the maximum number of allowed intervals for the certified key.

6.2.2 Adaptation of revocation mechanisms

CRLs and OCSP are the standard mechanisms to provide revocation information as described in Chapter 3. Thus, it must be possible to provide a revocation index within these data structures to enable fine grained revocation checking as described in Section 6.1.2.

In general, a revocation of a certificate is published as a CRL entry within a CRL or within an OCSP response. For the purpose to provide auxiliary data associated to a CRL entry or an OCSP response, CRL entry extensions have been defined [83]. These can be included in the `crlEntryExtensions` field as part of an CRL entry (cf. Listing 6.2) as well as into OCSP responses within the `singleExtensions` field as shown in Listing 6.3 [104].

```
CertificateList  ::=  SEQUENCE  {
        tbsCertList           TBSCertList,
        signatureAlgorithm    AlgorithmIdentifier,
        signatureValue        BIT STRING  }

   TBSCertList  ::=  SEQUENCE  {
        version                  Version OPTIONAL,
                                      -- if present, MUST be v2
        signature                AlgorithmIdentifier,
        issuer                   Name,
        thisUpdate               Time,
        nextUpdate               Time OPTIONAL,
        revokedCertificates      SEQUENCE OF SEQUENCE  {
             userCertificate          CertificateSerialNumber,
             revocationDate           Time,
             crlEntryExtensions       Extensions OPTIONAL
                                       -- if present, version MUST be v2
                                  }  OPTIONAL,
        crlExtensions            [0]  EXPLICIT Extensions OPTIONAL
                                       -- if present, version MUST be v2
                                  }
```

Listing 6.2: CRL [83]

```
SingleResponse ::= SEQUENCE {
   certID                       CertID,
   certStatus                   CertStatus,
   thisUpdate                   GeneralizedTime,
   nextUpdate          [0]      EXPLICIT GeneralizedTime OPTIONAL,
   singleExtensions    [1]      EXPLICIT Extensions OPTIONAL }
```

Listing 6.3: OCSP single response [104]

The definition of an according extension for the revocation index enables its X.509 standard conform transmission. Listing 6.4 shows an X.509 extension and the proposed definition of the `revocationIndex` extension. An extension is associated with an OID to identify it. Optimally, the OID should be a member of the id-ce arc (2.5.29) for ISO/ITU-T jointly assigned OIDs for certificate extensions. The number represented by the placeholder 'XXX' needs to be requested during the standardization procedure.

```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID      OBJECT IDENTIFIER,
     critical    BOOLEAN DEFAULT FALSE,
     extnValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the extension type identified
                 -- by extnID
     }

id-ce-revocationIndex OBJECT IDENTIFIER ::= { id-ce XXX }

RevocationIndex ::= INTEGER (0..MAX)
```

Listing 6.4: Revocation index extension

The `revocationIndex` extension is to be a non-critical extension in order to guarantee backward compatibility. A client not supporting the extension would then ignore it and consider the complete certificate as revoked.

The revocation index needs to be identified, when a certificate is to be revoked. Considering a revocation because of a CA compromise, the identification of the revocation index depends on the type of FSS. Either, the index depends directly on the date of the compromise, or on the index of the last signature which was legitimately generated with the compromised key. Therefore, the certificate owner has to keep track of his key updates. In case of a CA, this can be done by logging the last used index with the current date to a write once memory during key update. Then, given the date of the key compromise, it is possible to determine the last index used before the compromise.

6.2.3 Adaptation of path validation

Given the revocation index is provided along with a revocation, the path validation needs to be adapted such that the revocation index is evaluated during revocation checking.

The implementation of the validity model given in Definition 6.1 only requires minor changes of the path validation specified in RFC 5280 [83] in order to consider revocation indices. Revocation checking is part of Section `6.1.3. Basic Certificate Processing` in RFC 5280. According to the shell model, in step (a) (3), it is checked whether the certificate (called current certificate) is revoked at the current time or not. In order to realize the chain model for FFS with time limitation a differentiation between certificates that certify an FSS key and common certificates needs to be done. If the current certificate certifies an FSS key, it must be checked that the signature index extracted from the signature on the subsequent certificate in the path is smaller than a potential revocation index. Otherwise it must be checked that the certificate is not revoked at the current time.

This way, path validation is fully backward compatible and even certification paths where CAs do not exclusively use FSS can be processed. In particular, this is relevant when transitioning a PKI to a CA revocation tolerant PKI.

6.2.4 Deployment

For the deployment of a CA revocation tolerant PKI, the standardization of FSS is required in order to ensure interoperability. This must happen before CAs can start to employ FSS, as standardization is a preliminary for implementation on the client side. The client applications (e.g. web browsers) have to support the used FSS first. Otherwise clients become unable to verify certification paths once CAs employ the new schemes. Given standardization has happened, the verification algorithms of the FSS as well as the adapted path validation have to be implemented within major libraries and crypto providers, as e.g. OpenSSL [167] and NSS [155]. Once supported in major libraries, CAs can start to employ FSS for certificate issuance. On the side of web servers, changes are only required if client authentication is done based on certificates. Otherwise no changes are required due to the use of conventional signature schemes for end entities. This in particular means, that no changes in the TLS specification are required.

6.3 Evaluation

In the following, the proposed solution is evaluated regarding practicality, which includes the availability of an appropriate FSS. We show that the eXtended Merkle Signature Scheme (XMSS) [12], a hash-based FSS, fulfills the requirements regarding efficiency and security. Runtimes, key sizes and signature sizes are presented, which show that a CA revocation tolerant PKI realized with XMSS provides good performance and practical data loads in comparison to the standard setup with common signature schemes like RSA or (EC) DSA.

6.3.1 eXtended Merkle Signature Scheme (XMSS)

Although our proposal works with arbitrary FSS, we propose to apply XMSS. It is as fast as RSA and (EC) DSA although it is forward secure and it provides practical key sizes. These are the major requirements for a practical implementation of a CA revocation tolerant PKI. Furthermore, at the time of writing, the standardization process for XMSS has already been started [91, 130].

Besides that, XMSS comes with additional properties that bring additional benefits for the Web PKI besides solving the too-big-to-fail problem of CAs. We shortly summarize the properties of XMSS. It is hash-based and thus a post quantum signature scheme. Furthermore, XMSS can be realized with any secure hash function. For XMSS, an interval is hard linked to one single signature and the key is automatically updated by the signature algorithm. In case of a CA, this allows the most fine grained revocation handling that is possible.

Thus, the deployment of XMSS for the Web PKI brings the additional benefit that an alternative to RSA (which is currently used by more than 99% of all CAs for certificate signing [17]) is available once quantum computers are realized. Then, RSA as well as (EC) DSA become insecure and an alternative must be available anyway. Furthermore, XMSS brings the possibility to base security on different mathematical problems without requiring different signature schemes by using different hash functions for instantiation. This allows diversity without the additional standardization efforts for further signature schemes.

Furthermore, with XMSS the probability of a sudden breakdown caused by advances in cryptanalysis can be efficiently minimized [12]. On the one hand, it requires minimal security properties to be secure, thus the break of harder properties can be seen as an early-warning system. On the other hand, so-called hash combiners (i.e. see [19]) can be used, such that the resulting combination is secure as long as at least one of the hash function families is secure. Such a property is especially

relevant, when considering end entity signatures requiring long term verifiability as presented in Chapter 7.

The following evaluation is based on the efficiency of XMSS to show that the realization of a CA revocation tolerant PKI is practical.

6.3.2 Practicality of the CA revocation tolerant PKI

To evaluate the performance of our solution, it is analyzed, where the use of an FSS leads to computational overheads and additional data loads. Overheads may evolve from the use of the signature scheme itself and during the transmission of keys. Thus, we evaluate the performance of key generation, signature generation and verification. Concerning data loads, differences to the current setup may evolve from certificate sizes. The additional data resulting from the revocation index extension in CRLs and OCSP responses is considered negligible as it consists of a single integer value. The evaluation is based on [34], which provides an in depth analysis of the performance of XMSS and different parameter settings. An excerpt of timings and key sizes is presented in Table 6.1. The timings are measured with a C implementation, where XMSS is instantiated with hardware accelerated AES (AES-NI) using 128 bit keys. The bit security for XMSS deduced from known attacks is 128 bit in all cases. In brackets, a lower bound on the provable security is given. Note that such proofs do not exist for RSA and DSA. For RSA and DSA to reach a bit security of 128, a key length of 4440 bit is required according to the heuristic of Lenstra and Verheul with updated equations [48, 47]. A bit security of 128 is assumed to be secure until the year 2090, which shows that the chosen parameter setting is suitable to be used in the practice regarding security.

Signature generation and verification

For signature generation, and most important signature verification XMSS provides better timings than RSA2048 and DSA2048. This performance gain would even be stronger for longer key sizes in the case of RSA and DSA. Thus, XMSS provides a performance gain during path validation.

Limited number of signatures and key renewal

With XMSS the limited number of possible signatures has to be considered. However, while this would be a drawback when XMSS was used by end entities during a TLS handshake for authentication, in the case of certificate issuance this is no hindrance in practice.

	t	Timings (ms)			Sizes (byte)			
Alg.	(*1,000)	Keygen	Sign	Verify	Secret key	Public key	Signature	b
XMSS	1	55	0.24	0.07	804	596	2,292	128 (101)
XMSS	1	77	0.33	0.06	804	564	1,236	128 (88)
XMSS	65	3,505	0.41	0.07	1,332	788	2,388	128 (92)
XMSS	65	4,915	0.56	0.06	1,332	756	1,332	128 (82)
XMSS	1,000	56,066	0.52	0.07	1,684	916	2,452	128 (84)
XMSS	1,000	79,196	0.71	0.06	1,684	884	1,396	128 (78)
RSA 2048		-	3.08	0.09	$\leq$ 512	$\leq$512	$\leq$256	90-95
DSA 2048		-	0.89	1.06	$\leq$512	$\leq$512	$\leq$ 256	90-95

Table 6.1: XMSS performance for different parameter settings, instantiated with AES-NI. Measurements on a computer with an Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz and 8GB RAM. t denotes the number of possible signatures with one key pair. b denotes the bit security, numbers in parentheses denote a lower bound on the provable security. AES-NI is used with 128 bit keys. [34]

CA keys are exclusively used to sign certificates and CRLs. Thus, 1,000,000 possible signatures before a key renewal is required is considered sufficient. For example, one of the most active CAs in the certification business has signed around 90,000 TLS certificates directly [5]. Assuming a yearly renewal of TLS certificates and a weakly issuance of CRLs this implies that a CA can use its key pair for more than ten years. For Root CAs and Sub CAs that do not issue end entity certificates even one thousand signatures might suffice, as the number of Sub CAs a CA issues certificates for is normally limited to a few tenth. Note that an extension of XMSS, namely XMSS^{MT} is available, that allows for a virtually unlimited number of signatures [34].

Key generation timings are of limited relevance. XMSS key generation can be performed in less than 1.27 minutes as shown in Table 6.1. Taking into account, that this is only done once and it is an offline task, this does not interfere the use of XMSS.

Certificate sizes

Assuming the parameters are encoded within the OID that identifies the signature algorithm, the impact on certificate sizes by the employed signature scheme results from the subject public key and the CA's signature on the certificate. An XMSS public key is only included within CA certificates. Depending on the respective parameter setting, Table 6.1 shows that signature sizes are one to two kilobytes larger than those generated with RSA or DSA, while public key sizes are at most doubled.

Regarding certificate sizes this leads to a growth by 1-2.5 kilobytes. However, once key sizes of RSA and DSA need to be increased to 4096 bit, this disadvantage shrinks. We note that EC DSA would allow for shorter key lengths. However, less than 0.3% of the trusted CAs of the Web PKI make use of EC DSA keys [17]. Thus, EC DSA has no relevance for the comparison.

For an exemplary certification path that includes 2 Sub CAs, the data overhead during a TLS handshake due to certificate sizes is approximately 3,524 bytes. Note that the Root CA certificate needs not be transmitted during the handshake. This shows that the use of XMSS for certificate signing is practical. The average size of a web page listed in the Alexa Top 1 Million is 1.8 MB [9]. Thus, the additional data due to certificate sizes on average would make up for less than 0.2% of traffic overhead.

Hardware requirements

In [34], it has been shown that the use of XMSS does not require special hardware. The timings presented above have been measured on a standard computer, and even the applicability on a smart card has been shown.

The larger size of the private key in comparison to RSA or DSA is of minimal relevance. It is only to be stored and used on the CA's hardware security module, which is unproblematic for the key sizes given in Table 6.1.

Organizational changes

Required organizational changes is another aspect which is important for the deployment in practice. As the use of FSS for certificate signing simply requires the exchange of the used signature algorithm, no procedural changes for certificate requests and issuance are required. However, it should be noted that CAs must ensure the appropriate update of the private key according to the specification of the intervals of the key pair's lifetime. Using XMSS, this is ensured by an automatized update after each signature generation. However, as the correct key update is indispensable to achieve a CA revocation tolerant PKI deployment, this update requirement should be reflected in the CAs' policies and the approval of the implementation of adequate measures should become part of regular security audits.

6.3.3 Comparison to time-stamping

An alternate approach to distinguish between legitimately and maliciously issued certificates in the face of CA compromises is to add time-stamps by an independent

trusted TSA. The time-stamping approach has been proposed to implement the chain model in a secure way [6]. We shortly summarize, why time-stamps are not a practical solution for the too-big-to-fail problem.

To implement the chain model based on time-stamps, a time-stamp for any certificate in the certification path would be required [6]. With this, for each certificate it could be securely distinguished whether a certificate existed before a compromise and thus can be considered as legitimately issued. However, the approach has serious drawbacks and performance issues.

Firstly, the setup and maintenance of an additional and independent TSA infrastructure and the trustworthiness of the TSAs to apply the correct date and time is required. The time-stamping service may be provided by a CA, however this CA must be independent from the CAs in the certification path that is to be protected. Otherwise, a compromise that invalidates the certification path also invalidates the time-stamp. This means for each time-stamp in the certification path, an additional certification path has to be verified to authenticate the TSA's certificate. In the best case when all CAs use the same time-stamping service, at least one additional certification path has to be processed. However, this case is hardly imaginable without highly limiting the flexibility of the PKI because the time-stamp needs to be requested during or directly after the certificate issuance. This means, on the CA's side the certificate issuance processes would have to be adapted, as well as different CAs would have to cooperate in order to prevent different TSAs for the protection of one certification path.

Furthermore, besides the time-stamps, the additional certification path(s) have to be delivered to the clients for path validation. Thus, the amount of transmitted certificates as well as the efforts for revocation checking during the TLS handshake would at least be doubled. The additional provision of time-stamps and independent certification paths also requires the adaptation of current standards. Furthermore, the independent paths need to be processed by the clients, requiring the adaptation of path validation algorithms.

Finally, time-stamps relying on electronic signatures themselves face the same problems concerning compromise and expiration as common electronic signatures do. Time-stamps only defer the problem to the TSA infrastructure and do not solve it. That is, upon the compromise of a TSA or any superordinate CA, the issued time-stamps become invalid which then would require their renewal facing the same problems as certificate renewal in case of CA compromise. All together, we deduce that time-stamps are only a theoretical solution for the too-big-to-fail problem but not applicable in practice to realize a CA revocation tolerant PKI.

6.4 Conclusion

In this chapter, it was presented how to realize a CA revocation tolerant PKI using forward secure signature schemes. The CA revocation tolerant PKI preserves the validity of legitimately issued certificates in the face of CA certificate revocations. It prevents the undifferentiated invalidation of all certificates that rely on a revoked CA certificate and the associated unavailability of dependent web services. Thus, CA certificates can be revoked in case of a compromise to prevent misuse by a attacker. Moreover, due to the precise impact of revocation in the CA revocation tolerant PKI, a certificate can even be revoked on suspicion of a compromise. It was shown, that the Web PKI can be transitioned to a CA revocation tolerant PKI with minor efforts once forward secure signature schemes are standardized. No organizational changes are required, thus allowing a transition parallel to normal operation. All changes can be realized fully backward compatible and are covered by current standards. The proposal has been evaluated regarding practicality and performance. It was shown that with XMSS an appropriate FSS is available, which in fact is currently being standardized. XMSS allows the implementation of a CA revocation tolerant PKI without limitations. The only drawback is slightly increased data loads during the TLS handshake, while no special hardware is required and signature verification speed can even be increased. The usability and the computational effort to use the PKI services is equal to conventional signature schemes. Thus, the presented CA revocation tolerant PKI is a practical solution to the too-big-to-fail problem of the Web PKI.

7 Providing non-repudiation and long term verifiability

Public key infrastructures provide the possibility to verify the authenticity of public keys at the time the keys are being used. In the previous chapters, we have shown how to maintain this guarantee in practice. However, digital signatures as used today do not provide non-repudiation and long term verifiability. These properties require additional mechanisms.

Conventional signature schemes such as RSA and DSA cannot guarantee non-repudiation as there exists no possibility to distinguish between signatures generated by the legitimate key owner or an attacker that compromised the key. This fact can be exploited by the key owner to repudiate formerly generated signatures by pretending that his key has been compromised. Long term verifiability has two aspects. First it requires an alternate validity model, called the chain model. The second aspect is the preservation of the security of signatures. Signature schemes become insecure over time. Thus, in the future an attacker might be able to forge signatures without knowing the according key. At that point, all signatures generated with the affected signature scheme become insecure because of the indistinguishability of legitimate and forged signatures.

Today, both problems are solved with time-stamps generated by time-stamping authorities. Yet, this solution is costly and therefore has prevented the broad application of digital signatures as a replacement for handwritten signatures so far.

In Section 7.1, we present a solution based on FSS, adopting the mechanisms used in Chapter 6. Non-repudiation is achieved by preventing back dated revocation. To do so, we exploit the chronological ordering of signatures provided by FSS. The state of the signature key is securely tracked by a trusted third party. This approach additionally allows reconfirmations for signature generations, e.g. similar to mTAN as known from online banking, which prevent unnoticed key misuse. Long term verifiability is addressed in Section 7.2. Firstly, FSS allow the application of the chain model without time-stamps. The second aspect of long term verifiability

is addressed with the use of XMSS, a hash-based FSS. We present possibilities to prevent the sudden break down of the security of signatures based on special properties of XMSS.

In Section 7.3, our solution is evaluated. The correctness is shown with a formal PKI model. The efficiency is evaluated by comparing the presented solution to the time-stamping based one. We evaluate data loads, runtimes and security requirements in the different setups and show that the FSS based solution has clear advantages. Section 7.4 concludes this chapter.

The contributions of this chapter were published as parts of [B6, B7, B8, B10].

7.1 Guaranteeing non-repudation

In this section we show how to achieve non-repudiation with FSS. First, an introduction to the non-repudiation scenario is given and the difficulties are explained in Section 7.1.1. Afterward, it is shown how to generally apply FSS for end entities in Section 7.1.2 and the model for signature validation is presented in Section 7.1.3. To eventually be able to guarantee non-repudiation, the secure tracking of key states of end entities is required. This is solved with the Sign & Report approach presented in Section 7.1.4. Finally, it is shown in Section 7.1.5, how to extend the Sign & Report approach with a compromise detection mechanism using a reconfirmation procedure.

7.1.1 Non-repudiation – motivation and problems

Over the past few years, the importance of e-business and e-government has been steadily growing. More and more processes are handled online without physical interaction. To guarantee for authenticity and non-repudiation in such processes, digital signatures are used. Moreover, many countries allow to replace handwritten signatures by digital signatures and consider these as legally binding [106]. This theoretically allows to transfer many processes to the digital world that formerly required a media disruption, e.g. in many countries applying for a bank account. However, there are several hurdles that lead to a rather low adoption of digital signatures as a replacement for handwritten ones over the past years.

Other than in authentication scenarios as considered in the previous chapters where the validity of a signature is only to be checked once at the time of authentication, for use-cases that require non-repudiation the validity of a signature must be provable as long as the signature is of any interest. In many cases, non-repudiation

must be preserved for ten years and more by law, which also directly links to long term verifiability covered in Section 7.2.

Now, there is a fundamental difference between handwritten and digital signatures. While handwritten signatures are naturally bound to a single person, the binding between digital signatures and a person is artificial and thus fragile. This binding, which is provided by PKIs is only temporary, terminated either by expiry of the certificate or its revocation. And this is where the non-repudiation property, which is guaranteed by the digital signature in theory fails in reality if there are no additional measures. The private key, required to generate signatures, can be applied by anyone who knows it or has access to it, without any possibility to distinguish which signature has been generated by whom. And because in principle a compromise is possible, a key owner can simply claim that his key was compromised, ascribing the generation of signatures to an attacker and thereby repudiating valid signatures. To prevent such a repudiation attack, a provable chronological order of events is required and must be considered during signature validation. A signature should then be verified as valid, if it was generated before a key compromise.

In the following we show how to establish a provable chronological order of events and maintain non-repudiation based on FSS. The use of FSS enables us to get rid of the drawbacks connected to the current solution with time-stamps.

7.1.2 FSS for end entities

In Chapter 6, it was shown how to employ the chronological ordering of signatures given by FSS to prevent the invalidation of legitimately issued certificates upon a CA certificate revocation. In this section, this mechanism is applied for end entity signatures in order to prevent the invalidation of legitimately generated signatures upon revocation of an end entity certificate.

Basically, the extension of the mechanism is achieved by additionally replacing the conventional signature schemes used by end entities with FSS and realizing fine grained revocation for end entity certificates analogously to Section 6.1.2.

However, in the case of end entities, the issue of triggering the key update algorithm must be taken into account. Recall that on-time key updates are indispensable to preserve forward security. The key update algorithm can either be called manually by the user, scheduled to run at the end of a certain time period, or be part of the signature algorithm, depending on the way the intervals of the key pair's lifetime are defined.

For end entities, FSS where the periods are based on the number of signatures are to be used. FSS based on the number of generated signatures have the advantage

that key update can be performed automated, based on a counter contained in the key holding device. The drawback is, that the key indices are not linked to real time, which complicates correct revocation in practice. This is because the index at the time of compromise must be traceable. Yet, this is achievable as shown in Section 7.1.4.

In comparison, the key update problem of FSS where intervals are defined in terms of time periods makes their application problematic. For these schemes, where e.g., one time period corresponds to one day, the key update algorithm must be triggered periodically. This can only be automated on systems that have an internal clock and that are active each time an update is necessary. On smartcards, which are the common place to store end entity signature keys, a manual update is required and thus does not allow any guarantees for on-time key updates.

In the following we assume an FSS that evolves the key after each signature generation. With XMSS an efficient scheme of this type is available as shown in Chapter 6.

7.1.3 Adaptation of the validity model

Extending the use of FSS to the end entity case requires a slight adaptation of the validity model given in Definition 6.1 to also consider the end entity signature. The adapted version is given in Definition 7.1.

For the definition let $n \in \mathbb{N}$ be the length of the certification path $p = (C_1, \ldots, C_n)$. C_1 is the self-signed certificate of the Root CA. C_n is the certificate of the end entity. We denote by $T_i(k)$ the starting date of the validity period of C_k and by $T_e(k)$ its expiration date. T_v is the time of signature verification. Additionally, let I_s be the index used for end entity signature generation, let $I_s(k)$ be the signing index used to sign certificate C_k and $I_r(k)$ a possible revocation index for certificate C_k. Further let $I_r(k) = \infty$ in case there is no revocation for certificate C_k. Recall that $I_s(k)$ and $I_r(k-1)$ are indices belonging to the same key pair.

Definition 7.1 (Chain model for FFS with time limitation – signature validation)**.** *A digital signature with index I_s is valid at verification time T_v if:*

1. *C_n is valid at verification time T_v: $T_i(n) \leq T_v \leq T_e(n)$ and C_n is not revoked for I_s: $I_s < I_r(n)$.*

2. *Every CA certificate in the path is valid at the verification time T_v: $T_i(k) \leq T_v \leq T_e(k)$ and not revoked for the signing index $I_s(k+1)$ used for the subordinate certificate in this path: $I_s(k+1) < I_r(k)$ for all $1 \leq k \leq n-1$.*

This validity model still limits the validity of signatures to the validity periods of the involved certificates to guarantee the availability of revocation information. Thus, it does not allow long term verification which requires the time limitation to be dropped. This will be considered in Section 7.2.

The application of FSS for end entity signatures solves the unintended invalidation of legitimately generated signatures. However, the assumption of dishonest end entities aiming at repudiating their own signatures leads to an additional challenge. Namely how to guarantee the correctness of the revocation index. This is solved with the Sign & Report approach presented in the following section.

7.1.4 Sign & Report

With the use of FSS for end entity signatures and the validity model given in Definition 7.1 legitimate signatures are not invalidated by a revocation, given the revocation index is correct. In order to guarantee non-repudiation, the PKI must ensure the correctness of the revocation index. In particular, a PKI that offers non-repudiation *must not* allow back dated revocation (for a formal proof refer to Section 7.3.1). However, when considering the facets of back dated revocation there are different security goals that contradict each other. This conflict needs to be resolved, and is discussed in the following. Afterwards, we present the Sign & Report approach to effectively prevent back dated revocation.

Back dated revocation

There are certain scenarios that require back dated revocation. Namely, whenever it is possible that the signature key might get compromised and maliciously used without being noticed immediately by the key owner. For example, consider a classical setup for digital signatures where the private key is stored on the user's system (e.g., PC). Here, the detection of a key compromise may take some time in which the attacker who stole the key may already have generated signatures. Then, it is clearly impossible to prohibit back dated revocation because back dated revocation is required to invalidate the signatures generated by the attacker before the compromise detection.

However, as back dated revocation contradicts non-repudiation, scenarios with such a (possibly large) gray phase must be excluded or the gray phase must be eliminated by technical means. Therefore, the secret key has to be protected in a way that prevents unnoticed compromises. A common solution that allows for a minimal gray phase is to store keys on smartcards, trusted platform modules (TPM) etc. Private keys are not extractable from these devices and can only be used when

the according secret, such as a PIN, is known. This allows for the assumption of immediate key compromise detection and subsequently the prohibition of back dated revocation in the sense that an attacker is not able to immediately crack the additional secret and thus use the stolen key before the key compromise, i.e. disappearance of the key storage device, is detected. In Section 7.1.5 we describe how the Sign & Report approach can be extended to explicitly support the detection of illegitimate key uses. This further justifies the assumption of a marginal gray phase, thereby eliminating the need for back dated revocation.

Sign & Report

To prevent back dated revocation, the responsible CA must know the current index of an end entity's key pair and be able to verify its correctness. To achieve this, the Sign & Report approach for FSS is defined. The basic idea is that the current index is reported to the CA after signature generation. This procedure enables the CA to keep track of the signing index and prevent the key owner from back dated revocation and repudiation of signatures. The index reporting protocol is presented in the following. Recall that an FSS is applied that evolves the key after each signature generation. Thus, each signature is directly linked to a unique index.

Index reporting protocol The index can be reported either by the signer or the verifier. This might be chosen depending on the specific application. The first case is desirable when the verifier is offline. However, then the signer needs to be able to prove the reporting. This can be realized by a *validity token* obtained from the CA and additionally serving as proof for the absence of a revocation making additional revocation checking obsolete. In the second case, reporting can directly be performed in one combined step during online revocation status checking and would reflect the natural ambition of the verifier to obtain non-repudiation.

Figure 7.1 shows the protocol for the first case, but the adaptation to the second is straight forward. After signature generation (Step (a)) the signature σ is sent to the CA together with the message m and an identifier ID of the signer (Step (b)). The ID can in particular be the certificate serial number of the signer's certificate. Recall, that in general, the hash of the message $h(m)$ is signed instead of signing m directly. If the signature scheme applied by the signer uses this initial hashing only for compression, but it would also be secure to sign messages directly, then it suffices to send $h(m)$ to the CA in Step (b) instead of m. This conceals m from the CA and in general prevents data overhead.

The CA checks the signature for validity (Step(c)) and generates a validity token

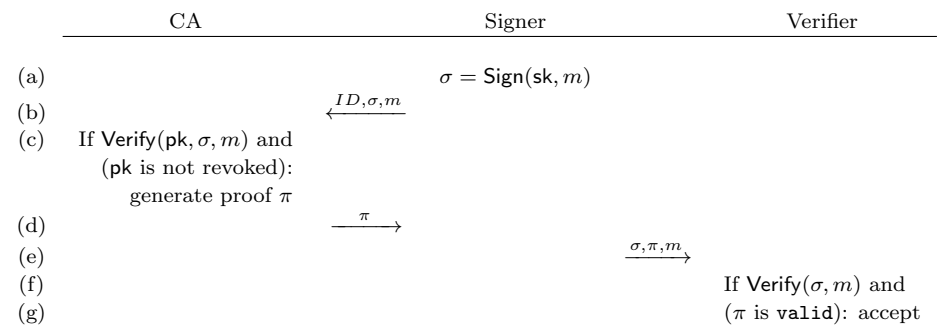


Figure 7.1: Index reporting protocol

π for the signature index contained in σ to confirm the logging. The signature verification ensures, that the signer's certificate is not revoked. Additionally, the reporting as well as the confirmation of wrong index information is prevented. π is sent back to the signer and subsequently transmitted (together with σ and m) to the verifier (Steps (d)-(e)), who can now validate the signature and the token.

By the index, the validity token is bound to a specific signature. Thus, it can be used for all future verifications without further online requests. Additionally, due to the forward security, the token for a certain index i can serve as a validity token for all preceding indices. Thus, if several signatures have to be validated, the logging request can be aggregated to only one, by requesting the token for the highest index. The validity token can be realized as (public key, index) pair signed by the CA. A convenient realization is shown in the following.

Integration into OCSP The index reporting can be integrated into OCSP. The OCSP standard allows extensions according to the extension model for X.509 certificates for OCSP requests as shown in Listing 7.1.

```
Request          ::=     SEQUENCE {
    reqCert                      CertID,
    singleRequestExtensions      [0] EXPLICIT Extensions OPTIONAL }
```

Listing 7.1: OCSP (single) request [104]

The definition of an appropriate extension allows the reporting of a new signature (index). The proposed definition of the `indexReport` extension is shown in Listing 7.2. The extension can then be included in the `singleRequestExtensions` field of an OCSP request.

```
id-pkix-ocsp-indexReport OBJECT IDENTIFIER ::= { id-pkix-ocsp XXX }

IndexReport ::= SEQUENCE {
     messageValue          BIT STRING,
     signatureValue        BIT STRING }
```

Listing 7.2: Signature index report extension

The extension is associated with an OID to identify it. Optimally, the OID should be a member of the id-pkix-ocsp arc (1.3.6.1.5.5.7.48.1) for ISO assigned OIDs for OCSP. The number represented by the placeholder 'XXX' needs to be requested during the standardization procedure, which is out of scope of this thesis. The extension contains the message m (in general represented by the hash value $h(m)$) in the `messageValue` field and the signature σ in the `signatureValue` field. The signature index is contained within σ. The certificate and thus the signer's public key is identified by the `CertID` contained in the OCSP request. With this information, the OCSP server can verify the signature and subsequently generate the validity token.

The validity token π is the standard OCSP response containing `good` as certificate status and the highest so far reported index for this certificate incremented by one in the `revocationIndex` extension defined in Listing 6.4.

Note that the OCSP server needs to implement the index logging functionality and requires access to the CA's certificate database. Furthermore, there can exist multiple OCSP responses for one certificate specifying different revocation indices as long as the response contains `good` as certificate status. Once the certificate is revoked, the OCSP response contains `revoked` as certificate status and the revocation index must not be changed anymore. However, even a response declaring the certificate as revoked can serve as a validity token for lower signature indices due to the forward security property of the signature scheme applied by the signer.

7.1.5 Incorporation of compromise detection

The Sign & Report approach makes it possible to monitor key usage and support end entities in the detection of illegitimate key usage and trigger revocation. Thus, the justification for immediate revocation can be strengthened. Compromise detection and the prevention of gray periods can be addressed by adding an additional reconfirmation procedure for signature generation. Before confirming the logging of the index, the CA can request a reconfirmation from the key owner. A possibility to do so is to apply mobile transaction numbers (mTAN) as commonly known from e-banking or similar to the usage presented in [B11].

The protocol flow with reconfirmation is shown in Figure 7.2. In Step (c) a random transaction number TAN is generated instead of the validity token π. The TAN is sent to some out of band device (OOBD) of the signer, e.g. its smartphone (Step (d.1)). The signer reads the TAN^* from the OOBD and then sends it back to the CA if he indeed used his private key and wants to report this to the CA (Steps (d.2) - (d.3)). Sending back the TAN^* reconfirms the will of the signer to report a new index. If TAN^* sent back by the signer equals TAN sent by the CA, then the CA generates the proof π and hands it to the signer (Steps (d.4) - (d.5)). The rest is analogue to the protocol shown in Figure 7.1.

As the signer is actively involved into the logging process and is informed about key usage via an independent channel, unintended key usage can be discovered. Thus, undetected usage of the key is significantly less probable, and even such cases can be detected, where e.g., the smartcard is left unwatched for a certain time span. Even reconfirmation of the actually signed message can be realized. By sending the message m in plain instead of its hash $h(m)$ to the CA, m could additionally be sent back and displayed on a smartphone for verification. A drawback of the reconfirmation procedure is that it cannot be integrated directly into OCSP, because OCSP is not designed for an interactive challenge response approach.

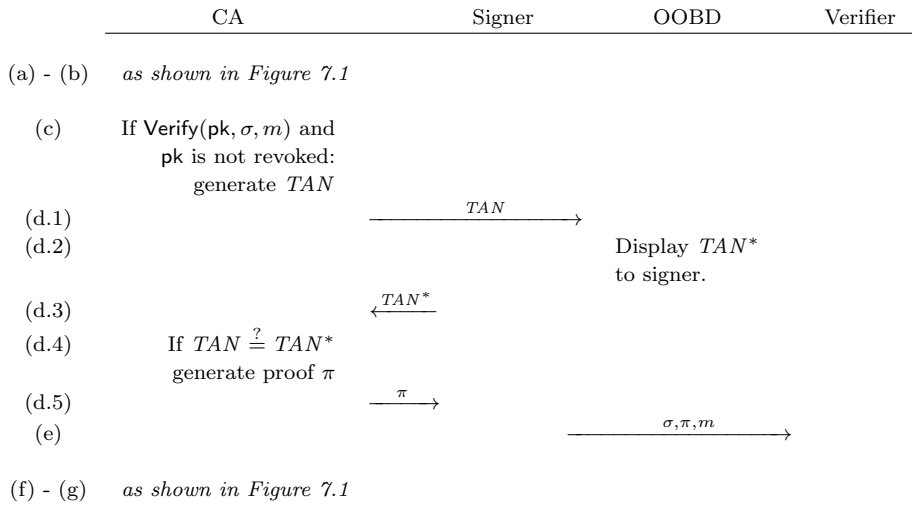


Figure 7.2: Index reporting with reconfirmation

7.2 Long term verifiability of end entity signatures

In this section, it is described how long term verifiability is achieved. First, in Section 7.2.1, it is explained how the time limitation of the validity model presented

in Definition 7.1 can be dropped in order to support long term verifiability. Then, in Section 7.2.2, the second aspect of long term verifiability – the preservation of the security of signatures – is addressed with the use of XMSS. We present possibilities to prevent the sudden break down of the security of signatures based on special properties of XMSS.

7.2.1 Chain model without time limitation

The chain model for FSS without time limitation is given in Definition 7.2. It allows the verification of a signature independent from the verification time, thus allowing long term verifiability. As mentioned in Section 7.1.3, the availability of revocation information is a particular challenge, when considering the validity of signatures upon the expiry of the certificates in the according certification path. So far, this availability was indirectly ensured by the time limitation. As this is not the case when the time limitation is dropped, the availability of revocation information is added as an explicit requirement.

For the definition let $n \in \mathbb{N}$ be the length of the certification path $p = (C_1, \ldots, C_n)$. C_1 is the self-signed certificate of the Root CA. C_n is the certificate of the end entity. T_v is the time of signature verification. Let I_s be the index used for end entity signature generation, let $I_s(k)$ be the signing index used to sign certificate C_k and $I_r(k)$ a possible revocation index for certificate C_k. Further let $I_r(k) = \infty$ in case there is no revocation for certificate C_k. Recall that $I_s(k)$ and $I_r(k-1)$ are indices belonging to the same key pair.

Definition 7.2 (Chain model for FSS – signature validation)**.** *A digital signature with index I_s is valid at verification time T_v if:*

1. *A valid revocation status is available for all certificates in the path p.*
2. *C_n is not revoked for I_s: $I_s < I_r(n)$.*
3. *Every CA certificate in the path is not revoked for the signing index $I_s(k+1)$ used for the subordinate certificate in this path: $I_s(k+1) < I_r(k)$ for all $1 \leq k \leq n-1$.*

Preserving revocation information

In order to allow long term verifiability valid revocation information must be available. In general it can be assumed that the initial verification of a signature is in close temporal proximity to the signature generation. As the verifier is always interested in obtaining a valid signature, he will never accept and store a signature

without validating it. Due to this temporal proximity, the initial validation can be performed according to Definition 7.1, thus guaranteeing the availability of revocation information. To prevent the possibility of `good` statements for non-issued certificates, non-repudiation scenarios should require that revocation information is obtained from OCSP servers that respond with `revoked` to requests for non-issued certificates. Such OCSP responders must include the `extended-revoke` extension according to RFC 6960 (cf. Section 4.4.8 in [104]). For the end entity signature, the validity token as defined in Section 7.1.4 is to be used. It directly states, that the signature in question has been reported to the CA and the according certificate is not revoked. Note that the signatures on the revocation information also need to be verified. Thus, in case the CA delegated the provision of revocation information, the according certificate must be stored in addition.

Then, the verifier can preserve the revocation information by storing it along with the obtained signature and the signed document. Thus, path validation according to Definition 7.2 is possible at any point in the future as long as the involved signature schemes are considered secure. This refers to the second aspect of long term verifiability and will be addressed in the following section.

7.2.2 Preventing the sudden break down of signature security

Having solved the theoretical aspect of long term verifiability with FSS, in this section we address the issue of signature schemes becoming insecure over time. This is especially relevant in scenarios, where the validity of signatures and the authenticity of stored data must be preserved over decades or even many generations. Many such scenarios exist, for example digital records in land registers, medical data or tax statements [B3, 66].

One major reason requiring preservation mechanisms for digital signatures is, that cryptographic algorithms underlie an aging process. Because of continually growing computational power and progress in cryptanalysis algorithms become weak over time. Thus, in the future an attacker might be able to forge signatures without knowing the according key. At that point, all signatures generated with the affected signature scheme become insecure because of the indistinguishability of legitimate and forged signatures. In practice, this aging process is counteracted with the adaptation of security parameters such as increasing the key lengths. The signatures that already have been generated and whose security needs to be preserved, have to be renewed whenever the employed parameters are about to become insecure. Renewal means, that existing signatures are signed together with the document by a trusted third party using a secure signature scheme and parameters. For this aspect,

a multitude of archival solutions exists that face this continuous aging of signature algorithms [66].

However, there is another aspect. Signature schemes can, as any cryptographic algorithm, suddenly become insecure. This happens, when unexpectedly an algorithm is found that solves the underlying mathematical problem in an efficient way. For example, once quantum computers can be built, all currently used signature algorithms become insecure because quantum computers can solve the discrete logarithm and the factorization problem [62]. In such a case not only the parameters need to be adapted, but the complete signature algorithm needs to be replaced. Even more severe, if such a break happens unexpectedly, there might not be enough time to maintain the validity of signatures with signature renewal.

With XMSS the probability of a sudden break down caused by advances in cryptanalysis can be efficiently minimized. Being a hash-based signature scheme, XMSS is constructed using hash function families as building blocks. In particular, the construction of XMSS requires a second-preimage resistant hash function and a pseudorandom function family. These two required properties are strictly weaker security assumptions than collision resistance as discussed in Section 2.1.1. However, collision resistance is normally required from a hash function in order to be considered as secure. Being a strictly harder security assumption, collision resistance of a hash function is normally broken before there are attacks against the second-preimage resistance or pseudorandomness. Thus on the one hand, the break of harder properties such as collision resistance can be seen as an early-warning system. Once an attack against collision resistance of an employed hash function is available, there is still time for signature renewal.

On the other hand, so-called hash combiners can be used to replace the used hash function family. Hash combiners use two families per property, such that the resulting combination is secure as long as at least one of the families is secure. There are folklore hash combiners for these properties (i.e. see [19]). Denote the message input of a function by m and the key input by k. Given two second-preimage resistant hash functions h_1, h_2, the Concatenation-Combiner $H_{||}^{h_1,h_2}(m) = (h_1(m)||h_2(m))$ is known to guarantee second-preimage resistance, as long as at least one of the used hash functions has this property. Similarly, for pseudorandom function families f_1, f_2 the XOR-Combiner $H_{\oplus}^{f_1,f_2}(k,m) = (f_1(k,m) \oplus f_2(k,m))$ is known to guarantee for pseudorandomness, as long as at least one of the used functions has this property. As these hash combiners themselves are (hash) functions, they can be easily plugged into XMSS. So there is no need to use two different signature schemes to base the security on two different mathematical problems. One only has to use two different (hash) functions, based on different problems or with different constructions.

With this approach, signatures stay secure even if one of the underlying primitives becomes insecure which keeps enough time for signature renewal. Furthermore, signature renewals have to be performed less often, only depending on the continuous development of computational power and not on the anticipation of potential breakthroughs in cryptography.

7.3 Evaluation

In this section we evaluate our solution. First, a formal PKI model is presented that allows to model revocation and formally define non-repudiation. Within this model, we prove the correctness of the Sign & Report approach and that it indeed guarantees non-repudiation. The practicality to replace contemporary signature schemes by XMSS has already been shown. Please refer to Section 6.3.2 for this issue. The over all efficiency of our solution is evaluated based on a comparison to the application of time-stamps, which is the current standard to provide non-repudiation and long term verifiability [85, 86]. Data loads, runtimes and security requirements in the different setups are evaluated and it is shown that the FSS based solution has clear advantages.

7.3.1 The Sign & Report approach provides non-repudiation

In the following it is shown, that Sign & Report provides non-repudiation. To show this, a new extension to the formal security model introduced by Maurer in [54] is presented. In the analysis, CAs are assumed to be trustworthy and non compromised. It is focused on non-repudiation, which is an issue concerning malicious end entities. Following this assumption, the model only considers relations starting from Sub CAs that sign end entity certificates. How to handle attacks against CAs such that these do not invalidate legitimate signatures was presented in Chapter 6 and integrated into the solution as presented in Section 7.1.

Formal PKI Model

The model by Maurer [54] was extended by Marchesini et al. [51] and Bicakci et al. [10]. The model presented here is built upon [51] as they introduce a smooth notion of how to handle time. We generalize their model in the sense that we do not depend on real time, but allow any indexing that admits a chronological ordering. This still includes the usage of real time information for indexing. While all former models are static, meaning they model one snapshot of a PKI, we introduce transitions between

snapshots of the PKI, making the model dynamic. Then, explicit definitions of revocation handling and end entity signatures are added. This allows to discuss non-repudiation using our model. Those parts of former models used to model a web of trust are dropped.

A PKI is modeled as View of a potential user at a specific time t. A user's View is a set of statements. We define six different statements. Trust expresses the trust in a (Sub) CA, obtained according to the higher hierarchy or by explicitly trusting this CA. Cert says that the user has seen an end entity certificate of the respective person. If a user has seen a certificate once, it remains in his view. The same holds for Signature and Revoc, which model that a user has seen a document signature or revocation information, respectively. Furthermore, there are two different Valid statements, which model that a user is convinced of the validity of an end entity's certificate $C_{\alpha,\beta,\gamma,\varepsilon}$ or document signature $S_{\zeta,\eta,\delta}$. These two Valid statements can be inferred from other statements, using inference rules defined later. As we allow transitions between views, every View is indexed with a time $t \in \mathbb{N}$. Note that indices used inside statements might be independent from the indices of the views. We write View^t for the View at time t and View if no specific t is needed.

Definition 7.3 (Statements)**.** *Let* CA *denote a (Sub) CA,* $\mathcal{E}$ *an end entity's identity,* D *a document and* $\mathcal{I}$ *a (time) interval. A* $\mathsf{View}^t = \{\mathsf{stmt}_1, \ldots, \mathsf{stmt}_n\}$ *at point in time* t *consists of* $n \in \mathbb{N}$ *statements* stmt_i*. There exist the following six statements:*

$\mathsf{Trust}(\mathtt{CA}, \mathcal{I})$ *denotes the belief that, during the interval* $\mathcal{I}$*,* CA *is trustworthy for issuing certificates, i.e. models the axiomatic trust in (Sub) CAs.*

$\mathsf{Cert}(\mathtt{CA}, \mathcal{E}, i, \mathcal{I})$ *denotes the fact that* CA *has issued a certificate for* $\mathcal{E}$ *at index* i*, which, during* $\mathcal{I}$*, binds* $\mathcal{E}$*'s public key to the certificate.*

$\mathsf{Signature}(\mathcal{E}, D, i)$ *denotes the fact that* $\mathcal{E}$ *has signed a document* D *at index* i*.*

$\mathsf{Revoc}(\mathtt{CA}, C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *denotes the fact that* CA *has revoked the certificate* $C_{\alpha,\beta,\gamma,\varepsilon}$*, represented by statement* $\mathsf{Cert}(\alpha, \beta, \gamma, \varepsilon)$*, at index* i*.*

$\mathsf{Valid}(C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *denotes the belief that certificate* $C_{\alpha,\beta,\gamma,\varepsilon}$ *is valid at evaluation index* i*.*

$\mathsf{Valid}(S_{\zeta,\eta,\delta})$ *denotes the belief that signature* $S_{\zeta,\eta,\delta}$*, represented by statement* $\mathsf{Signature}(\zeta, \eta, \delta)$*, is valid.*

A statement is `valid` if and only if it is in the View or can be derived from it using one of the inference rules defined below.

Signature Validation Definition 7.4 gives the inference rules used to validate signatures, i.e. derive `valid` for a Signature. The rules depend on the validity model used for certification path validation, and are according to the chain model given in Definition 7.2.

Definition 7.4 (Inference Rules)**.** *Statements can be derived from an existing* View^t *according to the following rules:*

Certificate Validity $\forall\ \mathtt{CA}, \mathcal{E}, i_r \leq i_v, i_c \in \mathcal{I}_1, i_v \in \mathcal{I}_2\ :\ \mathsf{Trust}(\mathtt{CA}, \mathcal{I}_1), \mathsf{Cert}(\mathtt{CA}, \mathcal{E}, i_c, \mathcal{I}_2), (\neg\mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_r)) \vdash \mathsf{Valid}(C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_v)$

Signature Validity $\forall\ \mathtt{CA}, \mathcal{E}, D, i_s \in \mathcal{I}_2\ :\ \mathsf{Valid}(C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_s), \mathsf{Signature}(\mathcal{E}, D, i_s) \vdash \mathsf{Valid}(S_{\mathcal{E},D,i_s})$

Dynamization of the model So far the model is static. To allow the definition of non-repudiation *transitions* between views are introduced. The transitions model that new information enters a user's View in form of certificates, signatures or revocation information. Besides that, a user might trust a new (Sub) CA.

Definition 7.5 (Time & Transitions)**.** *Let* View^t *be the* View *at time* t *and* $\mathsf{View}^t \xrightarrow{\mathtt{trans}} \mathsf{View}^{t+1}$ *denote the transition from* View^t *to* View^{t+1}*. Let* CA *denote a (Sub) CA,* $\mathcal{E}$ *an end entity's identity,* D *a document and* $\mathcal{I}$ *an interval. We allow the following four transitions between views:*

- $\mathsf{View}^t \xrightarrow{\mathsf{Sign}(\mathcal{E},D,i)} \mathsf{View}^{t+1}$ *adds* $\mathsf{Signature}(\mathcal{E}, D, i)$ *to* View.
- $\mathsf{View}^t \xrightarrow{\mathsf{issue}(\mathtt{CA},\mathcal{E},i,\mathcal{I})} \mathsf{View}^{t+1}$ *adds* $\mathsf{Cert}(\mathtt{CA}, \mathcal{E}, i, \mathcal{I})$ *to* View.
- $\mathsf{View}^t \xrightarrow{\mathsf{trust}(\mathtt{CA},\mathcal{I})} \mathsf{View}^{t+1}$ *adds* $\mathsf{Trust}(\mathtt{CA}, \mathcal{I})$ *to* View.
- $\mathsf{View}^t \xrightarrow{\mathsf{revoke}(\mathtt{CA},C_{\alpha,\beta,\gamma,\varepsilon},i)} \mathsf{View}^{t+1}$ *adds* $\mathsf{Revoc}(\mathtt{CA}, C_{\alpha,\beta,\gamma,\varepsilon}, i)$ *to* View.

Derived statements are temporary. After a transition between two views, the inference rules are used again, to obtain the full set of statements. With $\overline{\mathsf{View}}$ we denote the set of all statements that can be inferred from View. So, if $\mathsf{stmt} \in \overline{\mathsf{View}^t}$ it does not have to be the case that $\mathsf{stmt} \in \overline{\mathsf{View}^{t'}}$ for $t \neq t'$. For example, if a certificate gets revoked, Valid might be inferable beforehand but not after Revoc has been added to the View.

Non-repudiation and back dated revocation

Now, the classic non-repudiation definition [94] is given in the presented model. This allows a more precise analysis of repudiation adversaries.

Definition 7.6 (Non-repudiation)**.** *A PKI offers non-repudiation if the following implication is always true, even in presence of a malicious end entity that might sign arbitrary messages, request new certificates and ask any CA to revoke any of his certificates at anytime.*

$$\forall\ i,\ t \leq t'\ :\ \mathsf{Valid}(S_{\mathcal{E},D,i}) \in \overline{\mathsf{View}^{t}} \Rightarrow \mathsf{Valid}(S_{\mathcal{E},D,i}) \in \overline{\mathsf{View}^{t'}}.$$

We briefly discuss the implications of this definition. The left part of the implication – $\mathsf{Valid}(S_{\mathcal{E},D,i}) \in \overline{\mathsf{View}^{t}}$ – implies that

$$\{\mathsf{Signature}(\mathcal{E}, D, i), \mathsf{Trust}(\mathtt{CA}, \mathcal{I}_1), \mathsf{Cert}(\mathtt{CA}, \mathcal{E}, i_c, \mathcal{I}_2)\} \subseteq \overline{\mathsf{View}^{t}}$$

with $i_c \in \mathcal{I}_1,\ i \in \mathcal{I}_2$ according to the previously given inference rules and definitions. Furthermore, $\mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_r) \notin \overline{\mathsf{View}^{t}}$ for all $\mathcal{I}_2 \ni i_r \leq i$. In other words, three things must be in View^{t}: (i) trust in the certification authority $\mathtt{CA}$ that issued the end entity certificate for the document signing entity $\mathcal{E}$, (ii) the certificate of $\mathcal{E}$ that has been issued while $\mathtt{CA}$ has been trusted, (iii) a signature on the verified document D that has been issued by the end entity $\mathcal{E}$ while his certificate has been valid, i.e. was not revoked or expired. The right part of the implication only differs in the time of inference of the Valid statement. Thus, everything above must hold for all future points in time t'.

Accordingly, the goal of the repudiation attacker is to produce a valid document signature $\mathsf{Signature}(\mathcal{E}, D, i)$ such that there exists a point in time t' where the signature is verified as invalid, after it has been verified as valid. Therefore, we define back dated revocation and show, that its prevention implies non-repudiation and vice versa in the chain model.

Definition 7.7 (Back dated revocation)**.** *Let* View^{t} *be the* View *at time* t *and* View^{t+1} *denote the view after a transition. According to the revocation transition, back dated revocation is defined as:*
$\mathsf{View}^{t} \xrightarrow{\mathsf{revoke}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}}, i_r)} \mathsf{View}^{t+1}$*, if* $\exists\ \mathsf{View}^{t^*} \ni \mathsf{Valid}(S_{\mathcal{E},D,i_s})$*, with* $t^* \leq t \wedge i_s \geq i_r$*.*

Theorem 7.8 (Non-repudiation $\Leftrightarrow$ no back dated revocation)**.** *A PKI offers non-repudiation according to Definition 7.6 if and only if it does not allow back dated revocation according to Definition 7.7.*

Proof. $\Leftarrow$: If there was a successful repudiation attack, then there must exist two views $\overline{\mathsf{View}^t} \supseteq \{\mathsf{Valid}(S_{\mathcal{E},D,i_s}), \mathsf{Trust}(\mathtt{CA}, \mathcal{I}_1), \mathsf{Cert}(\mathtt{CA}, \mathcal{E}, i_c, \mathcal{I}_2)\}$ and $\overline{\mathsf{View}^{t'}} \supseteq \{\mathsf{Trust}(\mathtt{CA}, \mathcal{I}_1), \mathsf{Cert}(\mathtt{CA}, \mathcal{E}, i_c, \mathcal{I}_2), \mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_r)\}$, with $t \leq t', i_r \leq i_s$. As $\mathsf{Valid}(S_{\mathcal{E},D,i_s})$ is contained in $\overline{\mathsf{View}^t}$, it can not contain $\mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_r)$. Hence, $\mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_r)$ must have been added later, which exactly corresponds to Definition 7.7.
$\Rightarrow$: If the PKI allows back dated revocation, the attacker is allowed to ask CA to add $\mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}_2}, i_r)$ with $i_r \leq i_s$ to the $\mathsf{View}^{t'}$. □

The Sign & Report PKI provides non-repudiation

Now the Sign & Report PKI is defined. Then, it is shown that it provides non-repudiation.

Definition 7.9 (Sign & Report PKI)**.** *A Sign & Report PKI implements the model defined in Section 7.3.1 replacing the abstract indices and intervals as described above. Let* R *denote a trusted third party in the PKI, e.g. a CA, which is responsible (and exclusively able) to issue the revocation of an end entity $\mathcal{E}$'s certificate $C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}}$, when requested by $\mathcal{E}$. Whenever $\mathcal{E}$ generates a signature, the used key index i^* is reported to* R *that stores i^*. On input of revocation request by $\mathcal{E}$,* R *publishes* $\mathsf{Revoc}(\mathtt{CA}, C_{\mathtt{CA},\mathcal{E},i_c,\mathcal{I}}, i^* + 1)$.

We next show that a Sign & Report PKI provides non-repudiation, assumed that the index reporting is secure.

Theorem 7.10 (Sign & Report PKIs provide non-repudiation)**.** *A Sign & Report PKI as defined above provides non-repudiation according to Definition 7.6.*

Proof. If the index reporting is implemented in a secure way, i.e. it is not possible for an end entity to manipulate the reporting, back dated revocation is efficiently prevented. This is the case, because the index used for revocation is greater than any index used by this end entity before. The non-repudiation property follows from Theorem 7.8. □

7.3.2 Comparison to time-stamping

In this section we compare our solution to the common approach of time-stamping. In Section 6.3.3, it was already shown, that time-stamps are inadequate to implement a CA revocation tolerant PKI.

Therefore, the evaluation is focused on a time-stamping approach that only provides non-repudiation and long term verifiability, but no CA revocation tolerant PKI

capabilities. Thus, the time-stamping approach is therefore inferior to the solution presented in this chapter. During the evaluation, we assume that TSAs also use conventional signature schemes. Basically, a time-stamp is added to an end entity signature directly after signature creation. The time-stamp also includes the certification path of the end entity certificate. The certification path must be included in order to prove its existence at the time of signature creation, otherwise a later CA compromise would necessarily invalidate the signature. As the time-stamp shows the time of signature creation, back dated revocation of the end entity certificate can be prevented by including the current time into the revocation, thus achieving non-repudiation in the face of repudiation attackers. Note, that in general the approach enables the validation according to the extended shell model (cf. Section 2.2.4).

The main drawback of the time-stamping approach is, that the TSA must be independent from all CAs involved into the certification path in order not to be affected in case of a CA compromise. Thus, the setup and maintenance of an additional and independent TSA infrastructure and the trustworthiness of the TSAs to apply the correct date and time is required. This independent infrastructure is not required in the solution based on FSS, as the forward security guarantees for the validity in case of CA compromises. In Section 7.1.4, it has been shown that even the index reporting can conveniently be integrated into the OCSP infrastructure.

Considering the validation of signatures, the independence requirement implies that for the verification of time-stamp signatures at least one additional certification path has to be processed. This doubles the runtime for signature validation. The FSS based solution does not introduce additional overhead for signature validation as shown in Section 6.3.2. The verification of the validity token is included in revocation checking as shown in Section 7.1.4, which is necessary anyway.

During signature generation, an additional online request to the TSA to generate the time-stamp is required. Such an online request can be completely omitted in the FSS based solution, if the verifier reports the signature during revocation checking. If the index reporting is realized by the signer, the online request during signature validation can be saved because of the provision of the validity token. Besides that, in the TSA solution signer and verifier need to agree on a TSA which is trusted by the verifier, while the FSS based solution is covered by the anyway trusted CAs.

Considering the data that has to be stored for future signature validations it is comparable in both cases. The TSA approach requires the storage of the two signatures (the document signature and the time-stamp) as well as two certification paths and related revocation information. The FSS based solution comes with only one signature and one certification path however increased signature and certificate sizes (cf. Section 6.3.2). But, compared to the validity token the storage of the

time-stamp is critical. If it is lost, the proof of existence of the signature is lost and can only be renewed for a later point in time. For a validity token, this is not the case as it can be obtained anew by requesting the revocation information (at least as long the end entity certificate has not expired).

This also shows a further issue with the TSA approach. Time-stamps relying on digital signatures themselves face the same problems concerning compromise as common digital signatures do. Upon the compromise of a TSA or any superordinate CA, all issued time-stamps become invalid and the proof of existence is lost. Thus, requiring the renewal of time-stamps which can only be made at a later point in time if no additional measures had been taken to guarantee the legitimacy of the time-stamps themselves.

For the long term preservation of signatures beyond the time when the involved signature algorithms might become insecure both solutions require archival systems. We refer the reader to [66] for an overview on common solutions. Basically, common solutions also apply time-stamps with up to date signature schemes. Such archival solutions are not covered in this thesis, we only remind the reader that the application of XMSS can protect from the sudden and unexpected break down of signature security as presented in Section 7.2.2. This is especially relevant for defining the frequency of repeated time-stamping.

7.4 Conclusion

In this chapter, it was presented how to achieve the non-repudiation property and long term verifiability for end entity signatures. This was realized by extending the mechanisms presented in Chapter 6 for the implementation of a CA revocation tolerant PKI to the end entity case. FSS were employed for end entity signatures. Together with tracking key states of end entity keys, non-repudiation can be guaranteed without any need for an additional trusted third party. This makes the presented solution clearly superior to the time-stamping based solution. Other than this, our solution comes with virtually no overhead as even the tracking of key states can be integrated into the revocation checking. Furthermore, it allows for a convenient integration of an additional reconfirmation step to detect compromises of end entity keys and to improve the overall security. In the evaluation, the existing formal models for a PKI have been extended such that it became possible to describe the non-repudiation property. The model was then used to prove the correctness of the presented solution.

Long term verifiability is achieved as FSS support signature validation according

to the chain model in a natural way. Besides that, the use of XMSS allows to prevent the sudden break down of signature security by basing the security on different underlying mathematical problems. This helps to minimize efforts in scenarios were signatures have to be archived for indefinite time periods and provides protection against sudden advances in cryptanalysis.

8 Conclusion

In this thesis we introduced CA-TMS to realize user-centric CA trust management for the Web PKI. Further we have shown how to combine FSS with today's PKIs in order to first build a CA revocation tolerant PKI, and second to establish non-repudiation as an inherent guarantee provided by PKIs.

For our proposed solutions the efficiency and practicality has been evaluated. The results have shown that the solutions are ready to use. As non of the proposals requires fundamental changes in the PKI processes, they can be implemented parallel to normal operation. Furthermore, none of the solutions requires broad deployment to be effective. CA-TMS, for which we provided an open source implementation, being installed on a single computer already protects that relying entity from malicious CAs. The use of FSS within PKIs solely requires that client systems do support the FSS. For deployment in the Web PKI standardization of the FSS is a preliminary. However, the expected benefits justify this effort.

CA-TMS reduces the attack surface by more than 95% for an average relying entity, thus providing protection from attacks based on fraudulent certificates. Other than existing solutions, CA-TMS on the one hand is dynamic and adapts to the changing requirements of a relying entity using certificate reconfirmation if it lacks sufficient local information for decision making. On the other hand, reconfirmations are only required for a small fraction of a relying entity's TLS connections, namely for approximately 0.27% to 0.69%. This solves scalability issues and limits the overhead as well as possible delays due to reconfirmations. We have also shown, that the user-centric data collection enables continuous revocation monitoring where for all certificates relevant to a relying entity the revocation status is at least checked once a day. It turned out that the expected load on OCSP servers is in the same range compared to online revocation checking as used today. The introduced service providers are an optional extension to CA-TMS. While providing valuable additions, these service providers are no preliminary for CA-TMS to work. By the realization of a reputation system, bootstrapping can be speed up by more than 50%. The individuality of trust views is preserved such that the use of the reputation system

only leads to a minimal increase of the attack surface by around 1%. Additionally, we have presented a push service for CA warnings to deal with behavioral changes. The related detection mechanism exploits the individuality of trust views and it has been shown to be highly efficient. Even if 90% of the members of the group of attacked entities trust the CA issuing the fraudulent certificates, on average this still leads to a detection after 9 attacked entities. We remind the reader, that in our test group of 64 relying entities, not a single CA achieved to be trusted by 90% of the entities.

Further we have shown how to build a CA revocation tolerant PKI by the application of FSS. The CA revocation tolerant PKI preserves the validity of legitimately issued certificates in the face of CA certificate revocations. Thus, CA certificate revocations need not be delayed but CA certificates can even be revoked on suspicion of a compromise. A secure state of the PKI can be reestablished immediately once a threat through a misbehaving CA or a CA failure has been detected. It was shown, how to transition the Web PKI into a CA revocation tolerant PKI. The performance analysis backs the claim that the proposed solution is practical and can be implemented without limitations.

The mechanisms of the CA revocation tolerant PKI have been adopted for scenarios where non-repudiation is required. Non-repudiation was established as an inherent guarantee provided by the PKI. This guarantee is preserved as long as the used signature scheme is considered secure. This makes time-stamps obsolete, thus saving the related overhead during signature generation and verification. The need for time-stamps can be postponed until the natural aging of the signature scheme makes a signature renewal necessary. In order to protect from unforeseeable developments and the sudden break down of signature security we have presented solutions based on hash-combiners, given XMSS is used as signature scheme. This ensures, that there is always enough time for signature renewal in scenarios were long term verifiability is required beyond the lifetime of the signature scheme itself.

Future work Still, there are several challenges and interesting research topics related to this thesis. One refers to the use of validation services for certificate reconfirmations, which is currently solved by the use of certificate notaries. While working good and in combination with CA-TMS the load on these services can be limited to face scalability problems, there is one major drawback. Such services must be operated by some third party which will always limit the number of such services. However, once such systems are not niche solutions anymore, they themselves might get the target of attacks. A more robust solution would be to integrate validation services as an integral component of the web infrastructure. A first attempt into

this direction has already been taken in [B1], where ubiquitous support of multi path probing was proposed. Having millions of potential servers providing means to reconfirm a certificate would significantly harden the system against attacks.

Also, compromise detection still is an issue and will be subject to future research. While the proposed service providers have been shown to enable this, the information must be fed back into the PKI in order to trigger revocations. Such processes should be automatized. Furthermore, we think that means for certificate owners to check their certificates from the outside are very important additions to a PKI. Again, multi path probing, and public logs like Certificate Transparency can be good starting points for future research.

Bibliography

[1] M. Abadi, A. Birrell, I. Mironov, T. Wobber, and Y. Xie. Global authentication in an untrustworthy world. *Proceedings of the 14th USENIX Conference on Hot Topics in Operating Systems (HotOS'13)*. USENIX Association, 2013. Cited on page 38.

[2] D. Akhawe and A. P. Felt. Alice in Warningland: A Large-scale Field Study of Browser Security Warning Effectiveness. *Proceedings of the 22Nd USENIX Conference on Security (SEC'13)*, pages 257–272. USENIX Association, 2013. Cited on page 50.

[3] M. Alicherry and A. D. Keromytis. Doublecheck: Multi-path verification against man-in-the-middle attacks. *14th IEEE Symposium on Computers and Communications (ISCC 2009)*, pages 557–563. IEEE Computer Society, 2009. Cited on page 52.

[4] R. Anderson. Two remarks on public key cryptology. *Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security (CCS)*, pages 1–4. Citeseer, 1997. Cited on page 12.

[5] H. Asghari, M. van Eeten, A Arnbak, and N. van Eijk. Security Economics in the HTTPS Value Chain. Social Science Research Network (SSRN), 2013. Cited on pages 35, 39, 40, 42, 43, 46, and 146.

[6] H. Baier and V. Karatsiolis. Validity models of electronic signatures and their enforcement in practice. *Proceedings of the 6th European conference on Public key infrastructures, services and applications - EuroPKI 2009*, pages 255–270. Springer, 2010. Cited on pages 18, 19, 21, and 148.

[7] R. L. Barnes. DANE: Taking TLS Authentication to the Next Level Using DNSSEC. *The IETF Journal*, vol.7(2), pages 4–7. Internet Society, 2011. Cited on page 53.

[8] M. Bellare and S. Miner. A forward-secure digital signature scheme. *Advances in Cryptology – CRYPTO' 99*, vol. 1666 of LNCS, pages 786–786. Springer, 1999. Cited on page 12.

[9] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. Cryptology ePrint Archive, Report 2014/795, 2014. Cited on page 147.

[10] K. Bicakci, B. Crispo, and A. S. Tanenbaum. How to incorporate revocation status information into the trust metrics for public-key certification. *Proceedings of the 2005 ACM symposium on Applied computing (SAC'05)*, pages 1594–1598. ACM, 2005. Cited on page 163.

[11] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society, 1996. Cited on page 29.

[12] J. Buchmann, E. Dahmen, and A. Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. *Proceedings of the 4th international conference on Post-Quantum Cryptography (PQCrypto'11)*, pages 117–129. Springer, 2011. Cited on pages 7, 135, and 144.

[13] J. Buchmann, E. Karatsiolis, and A. Wiesmaier. *Introduction to Public Key Infrastructures*. Springer, 2013. Cited on pages 13, 15, and 16.

[14] C. Burnett, T. J. Normal, and K. Sycara. Bootstrapping Trust Evaluations Through Stereotypes. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 (AAMAS '10)*, pages 241–248. International Foundation for Autonomous Agents and Multiagent Systems, 2010. Cited on page 71.

[15] D. W. Chadwick and A. Basden. Evaluating trust in a public key certification authority. *Computers & Security*, 20(7):592–611, 2001. Cited on page 29.

[16] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 2009. Cited on page 108.

[17] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC '13)*, pages 291–304. ACM, 2013. Cited on pages 3, 28, 36, 95, 99, 106, 144, and 147.

[18] C. Ellison and B. Schneier. Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. *Computer Security Journal*, 16(1):1–7, 2000. Cited on page 28.

[19] M. Fischlin, A. Lehmann, and K. Pietrzak. Robust multi-property combiners for hash functions revisited. *Automata, Languages and Programming*, vol. 5126 of LNCS, pages 655–666. Springer, 2008. Cited on pages 144 and 162.

[20] D. Gambetta. Can we trust trust? *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, 1988. Cited on page 24.

[21] I. Gassko, P. Gemmell, and P. MacKenzie. Efficient and fresh certification. *Public Key Cryptography*, pages 342–353. Springer, 2000. Cited on page 29.

[22] P. Giuli, P. Maniatis, T. Giuli, and M. Baker. Enabling the Long-Term Archival of Signed Documents through Time Stamping. *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST '02)*, pages 31–46. USENIX Association, 2002. Cited on page 29.

[23] M. Gleser. Policy-basierter Vergleich von Zertifizierungspraktiken innerhalb der Web-PKI. Bachelor thesis, TU Darmstadt, 2013. Cited on pages 38, 67, and 72.

[24] H.-w. Go. Forward security and certificate management in mobile AD Hoc networks. Master thesis, University of Hong Kong, 2004. Cited on page 30.

[25] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. Cited on page 11.

[26] P. Gutman. PKI: It's not dead, just resting. *Computer*, 35(8):41–49, 2002. Cited on page 28.

[27] P. Gutman. *Engineering Security*. Draft, 2013. Book draft available online at `http://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf`. Cited on pages 28, 42, and 43.

[28] S. M. Habib, S. Ries, S. Hauke, and M. Mühlhäuser. Fusion of opinions under uncertainty and conflict – application to trust assessment for cloud marketplaces. *TrustCom 2012*, pages 109–118, 2012. Cited on page 27.

[29] 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. Cited on page 111.

[30] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. *Proceedings of the 2009 workshop on New security paradigms workshop (NSPW '09)*, pages 133–144. ACM, 2009. Cited on pages 50 and 67.

[31] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1–31, Dec. 2009. Cited on pages 29, 90, and 118.

[32] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL landscape: a thorough analysis of the x.509 PKI using active and passive measurements. *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (IMC '11)*, pages 427–444. ACM, 2011. Cited on pages 28 and 36.

[33] J. Huang and D. Nicol. A calculus of trust and its application to PKI and identity management. *IDTrust '09*, pages 23–37. ACM, 2009. Cited on page 28.

[34] A. Hülsing. *Practical Forward Secure Signatures using Minimal Security Assumptions*. PhD thesis, TU Darmstadt, Sept. 2013. Cited on pages 10, 11, 145, 146, and 147.

[35] A. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010. Cited on page 112.

[36] A. Jøsang. An algebra for assessing trust in certification chains. *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'99)*. The Internet Society, 1999. Cited on pages 28 and 68.

[37] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9:279–311, 2001. Cited on page 25.

[38] A. Jøsang. Fission of opinions in subjective logic. *12th International Conference on Information Fusion (FUSION '09)*, pages 1911–1918. IEEE Computer Society, 2009. Cited on pages 25 and 27.

[39] A. Jøsang and R. Ismail. The beta reputation system. *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002. Cited on page 25.

[40] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43:618–644, 2007. Cited on pages 24 and 25.

[41] J. Kasten, E. Wustrow, and J. Halderman. CAge: Taming Certificate Authorities by Inferring Restricted Scopes. *Financial Cryptography and Data Security*, vol. 7859 of LNCS, pages 329–337. Springer, 2013. Cited on pages 94, 95, and 108.

[42] B. Kim, K. Choi, and D. Lee. Disaster Coverable PKI Model Utilizing the Existing PKI Structure. *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, vol. 4277 of LNCS, pages 537–545. Springer, 2006. Cited on page 29.

[43] S. Koga and K. Sakurai. Decentralization Methods of Certification Authority Using the Digital Signature Scheme. *2nd Annual PKI Research Workshop – Pre-Proceedings*, pages 54–64, 2003. Cited on page 30.

[44] L. Kohnfelder. *Towards a practical public-key cryptosystem.* PhD thesis, Massachusetts Institute of Technology, 1978. Cited on page 13.

[45] Z. Le, Y. Ouyang, J. Ford, and F. Makedon. A hierarchical key-insulated signature scheme in the ca trust model. *Information Security*, vol. 3225 of LNCS, pages 280–291. Springer, 2004. Cited on page 30.

[46] Z. Le, Y. Ouyang, Y. Xu, J. Ford, and F. Makedon. Preventing unofficial information propagation. *Information and Communications Security*, vol. 4861 of LNCS, pages 113–125. Springer, 2007. Cited on page 30.

[47] A. K. Lenstra. Key Length, 2004. Contribution to The Handbook of Information Security. Cited on page 145.

[48] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14:255–293, 2001. Cited on page 145.

[49] W. Lian, E. Rescorla, H. Shacham, and S. Savage. Measuring the Practical Impact of DNSSEC Deployment. *Proceedings of the 22th USENIX Security Symposium 2013*, pages 573–588. USENIX Association, 2013. Cited on page 53.

[50] S. Magin and S. Hauke. Towards engineering trust systems: Template-based, component-oriented assembly. *Proceedings of the Eleventh Annual International Conference on Privacy, Security and Trust (PST2013)*, 2013. Cited on page 29.

[51] J. Marchesini and S. Smith. Modeling public key infrastructures in the real world. *Proceedings of the Second European conference on Public Key Infrastructure (EuroPKI 2005)*, pages 118–134. Springer, 2005. Cited on page 163.

[52] S. Maseberg. *Fail-Safe-Konzept für Public-Key-Infrastrukturen*. PhD thesis, TU Darmstadt, 2002. Cited on page 55.

[53] M.-E. Maurer, A. D. Luca, and S. Kempe. Using data type based security alert dialogs to raise online security awareness. *SOUPS*, page 2, 2011. Cited on page 75.

[54] U. M. Maurer. Modelling a Public-Key Infrastructure. *Proceedings of the 4th European Symposium on Research in Computer Security: Computer Security (ESORICS'96)*, pages 325–350. Springer, 1996. Cited on pages 68 and 163.

[55] D. H. Mcknight and N. L. Chervany. The meanings of trust. Technical report, University of Minnesota, 1996. Cited on page 24.

[56] F. F.-H. Nah. A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & IT*, 23(3):153–163, 2004. Cited on pages 39 and 68.

[57] S. Ries. Extending Bayesian Trust Models Regarding Context-Dependence and User Friendly Representation. *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1294–1301. ACM, 2009. Cited on page 25.

[58] S. Ries, S. M. Habib, M. Mühlhäuser, and V. Varadharajan. CertainLogic: A Logic for Modeling Trust and Uncertainty (Short Paper). *TRUST 2011*, pages 254–261. Springer, 2011. Cited on pages 25 and 26.

[59] R. L. Rivest. Can We Eliminate Certificate Revocations Lists? *Proceedings of the Second International Conference on Financial Cryptography (FC '98)*, pages 178–183, 1998. Cited on page 29.

[60] S. Ruohomaa, L. Kutvonen, and E. Koutrouli. Reputation management survey. *Seventh International Conference on Availability, Reliability and Security (ARES 2007)*, pages 103–111, 2007. Cited on pages 25 and 29.

[61] G. Rynkowski. Individuell angepasstes Vertrauen in die Web-PKI: Eine Analyse des Einsparungspotentials an vertrauenswürdigen CAs. Master thesis, TU Darmstadt, 2013. Cited on page 57.

[62] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.*, 26:1484–1509, October 1997. Cited on page 162.

[63] C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against SSL. Technical report, Indiana University Bloomington - Center for Applied Cybersecurity Research, 2010. Cited on pages 28, 36, 40, 45, and 48.

[64] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. *Proceedings of the 18th Conference on USENIX Security Symposium*, 2009. Cited on pages 50 and 67.

[65] V. Tzvetkov. Disaster coverable PKI model based on majority trust principle. *International Conference on Information Technology: Coding and Computing*, 2:118, 2004. Cited on page 29.

[66] M. Vigil, J. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier. Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: a survey. *Computers & Security*, 50(0):16 – 32, 2015. Cited on pages 22, 161, 162, and 169.

[67] F. Volk and M. Mühlhäuser. Combining Inter-Component Rating and Performance Functions for Composite Web Service QoS Evaluation. *IEEE Transactions on Services Computing, Special Issue on emerging Web Services*, 2014. to be published. Cited on pages 113 and 114.

[68] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3 (NSDI'06)*, pages 1–1. USENIX Association, 2006. Cited on page 120.

[69] A. S. Wazan, R. Laborde, F. Barrère, and A. Benzekri. A formal model of trust for calculating the quality of x.509 certificate. *Security and Communication Networks*, 4(6):651–665, 2011. Cited on pages 29, 71, and 73.

[70] A. S. Wazan, R. Laborde, F. Barrère, and A. Benzekri. The x.509 trust model needs a technical and legal expert. *ICC*, pages 6895–6900, 2012. Cited on pages 29 and 71.

[71] G. A. Weaver, S. Rea, and S. W. Smith. A computational framework for certificate policy operations. *Public Key Infrastructures, Services and Applications - EuroPKI 2010*, vol. 6391 of LNCS, pages 17–33. Springer, 2010. Cited on page 29.

[72] S. Xu and M. Yung. Expecting the unexpected: Towards robust credential infrastructure. *Financial Cryptography*, vol. 5628 of LNCS, pages 201–221. Springer, 2009. Cited on page 30.

[73] Y. Yang, Y. Sun, S. Kay, and Q. Yang. Securing rating aggregation systems using statistical detectors and trust. *IEEE Transactions on Information Forensics and Security*, 4(4):883–898, 2009. Cited on page 120.

[74] Y. Zhang, J. I. Hong, and L. F. Cranor. Cantina: a content-based approach to detecting phishing web sites. *Proceedings of the 16th international conference on World Wide Web (WWW '07)*, pages 639–648. ACM, 2007. Cited on page 75.

[75] P. R. Zimmermann. *The official PGP user's guide*. MIT Press, 1995. Cited on pages 13, 28, 68, and 75.

Standards and RFCs:

[76] D. Eastlake 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Proposed Standard), Jan. 2011. Cited on page 18.

[77] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard), Aug. 2001. Updated by RFC 5816. Cited on page 22.

[78] ANSI X9.95-2012 — Trusted Time Stamp Management and Security, 2012. Cited on page 22.

[79] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), Mar. 2005. Updated by RFCs 6014, 6840. Cited on page 23.

[80] L. Bassham, W. Polk, and R. Housley. Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3279 (Proposed Standard), Apr. 2002. Updated by RFCs 4055, 4491, 5480, 5758. Cited on pages 139 and 141.

[81] CA/Browser Forum. Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates (v.1.2.3), 2014. Cited on pages 36, 46, and 139.

[82] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 3647 (Informational), Nov. 2003. Cited on page 21.

[83] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008. Updated by RFC 6818. Cited on pages 15, 16, 19, 35, 38, 138, 139, 140, 141, and 143.

[84] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176. Cited on page 23.

[85] ETSI. Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES), 2009. TS 101 733 V1.8.1. Cited on page 163.

[86] ETSI. Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES), 2009. TS 101 903, V1.4.1. Cited on page 163.

[87] C. Evans, C. Palmer, and R. Sleevi. Public Key Pinning Extension for HTTP draft-ietf-websec-key-pinning-21. IETF Internet-Draft, Oct. 2014. Cited on pages 49 and 50.

[88] A. Freier, P. Karlton, and P. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Historic), Aug. 2011. Cited on page 23.

[89] P. Hallam-Baker. X.509v3 TLS Feature Extension draft-hallambaker-tlsfeature-07. IETF Internet-Draft, Mar. 2015. Cited on page 54.

[90] P. Hoffman and J. Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698 (Proposed Standard), Aug. 2012. Cited on page 53.

[91] A. Hülsing, D. Butin and S. Gazdag. XMSS: Extended Hash-Based Signatures draft-xmss-00. IETF Internet-Draft, Mar. 2015. Cited on page 144.

[92] International Organization for Standardization (ISO). `http://www.iso.org`. Cited on page 22.

[93] International Telecommunication Union (ITU-T). `http://www.itu.int`. Cited on page 22.

[94] ISO/IEC 13888-1 — Information technology – Security techniques – Non-repudiation, Part 1, 2009. Cited on page 166.

[95] ISO/IEC 18014 — Information technology – Security techniques – Time-stamping services, 2009. Cited on page 22.

[96] ITU-T. Recommendation X.680 - Information Technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation, July 2002. Cited on pages 15 and 139.

[97] ITU-T. Recommendation X.509 (08/2005) - Information Technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. Technical report, ITU-T, Aug. 2005. Cited on page 15.

[98] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962 (Experimental), June 2013. Cited on page 52.

[99] S. Leontiev and D. Shefanovski. Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 4491 (Proposed Standard), May 2006. Cited on page 139.

[100] M. Marlinspike and T. Perrin. Trust Assertions for Certificate Keys draft-perrin-tls-tack-02.txt. IETF Internet-Draft, Jan. 2013. Cited on page 50.

[101] D. McGrew and M. Curcio. Hash-Based Signatures draft-mcgrew-hash-sigs-02. IETF Internet-Draft, July 2014. Cited on page 140.

[102] P. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Internet Standard), Nov. 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936. Cited on page 22.

[103] Y. Pettersen. The Transport Layer Security (TLS) Multiple Certificate Status Request Extension. RFC 6961 (Proposed Standard), June 2013. Cited on page 18.

[104] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960 (Proposed Standard), June 2013. Cited on pages 16, 17, 141, 142, 157, and 161.

[105] J. Schaad, B. Kaliski, and R. Housley. Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 4055 (Proposed Standard), June 2005. Updated by RFC 5756. Cited on page 139.

[106] The European Parliament and the Council of the European Union. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. *European Union law*. The European Parliament and Council, 1999. Cited on page 152.

[107] Web PKI OPS (wpkops) - IETF Working Group. Charter for Working Group. `http://datatracker.ietf.org/wg/wpkops/charter`. Cited on page 35.

Online resources:

[108] R. Alden. Web Browsers and Comodo Announce A Successful Certificate Authority Attack, Perhaps From Iran. Comment on Google Groups: mozilla.dev.security.policy, 29.03.2011. `https://groups.google.com/d/msg/mozilla.dev.security.policy/zgKmHOTIxn8/5NNYcgPNqlgJ`. Cited on page 43.

[109] J. Applebaum. Detecting Certificate Authority compromises and web browser collusion. Tor Project Blog, 22.03.2011. `https://blog.torproject.org/blog/detecting-certificate-authority-compromises-and-web-browser-collusion`. Cited on page 43.

[110] Multiple authors. Unbelievable! Discussion on Google Groups: mozilla.dev.tech.crypto, 22.12.2008. `https://groups.google.com/forum/#!topic/mozilla.dev.tech.crypto/nAzIKSBEh78[1-25-false]`. Cited on pages 42 and 43.

[111] K. Baumgartner. TURKTRUST CA Problems. SECURELIST Blog, 04.01.2013. https://www.securelist.com/en/blog/208194063/TURKTRUST_CA_Problems. Cited on page 46.

[112] N. Bolyard. Bug 470897 - Investigate incident with CA that allegedly issued bogus cert for www.mozilla.com. Bugzilla@Mozilla, 22.12.2008. https://bugzilla.mozilla.org/show_bug.cgi?id=470897. Cited on page 42.

[113] C. Brook. Deep Packet Inspection Firm Cyberoam Issues Fix Following Private Key Leak. threatpost, 09.07.2012. http://threatpost.com/deep-packet-inspection-firm-cyberoam-issues-fix-following-private-key-leak-070912/76779. Cited on page 46.

[114] CAcert Wiki. History of Risks & Threat Events to CAs and PKI. http://wiki.cacert.org/Risk/History. Cited on page 42.

[115] Carnegie Mellon University. Perspectives Project. http://perspectives-project.org. Cited on pages 51 and 88.

[116] chromium.org. Revision 108479. Chromium Update, 03.11.2011. https://src.chromium.org/viewvc/chrome?revision=108479&view=revision. Cited on page 46.

[117] Comodo. Comodo Report of Incident - Comodo detected and thwarted an intrusion on 26-MAR-2011. Incident Report, 31.03.2011. http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html. Cited on page 43.

[118] Cyberoam. Customer Knowledge Post - SSL Bridging, Cyberoam's Approach. Cyberoam Blog, 05.07.2012. http://www.cyberoam.com/blog/ssl-bridging-cyberoam-approach/. Cited on page 46.

[119] R. Duncan. Certificate Authorities struggle to comply with Baseline Requirements. Netcraft News Blog, 23.09.2013. http://news.netcraft.com/archives/2013/09/23/certificate-authorities-struggle-to-comply-with-baseline-requirements.html. Cited on page 67.

[120] P. Eckersley. EFF to Verizon: Etisalat Certificate Authority Threatens Web Security. EFFF Deeplinks Blog, 13.08.2010. https://www.eff.org/deeplinks/2010/08/open-letter-verizon. Cited on page 46.

[121] P. Eckersley. How secure is HTTPS today? How often is it attacked? EFF Deeplinks Blog, 25.10.2011. https://www.eff.org/deeplinks/2011/10/how-secure-https-today. Cited on page 48.

[122] P. Eckersley. Iranian hackers obtain fraudulent HTTPS certificates: How close to a Web security meltdown did we get? EFF Deeplinks Blog, 23.03.2011. https://www.eff.org/deeplinks/2011/03/iranian-hackers-obtain-fraudulent-https. Cited on page 43.

[123] Electronic Frontier Foundation. The Sovereign Keys Project. https://www.eff.org/de/sovereign-keys. Cited on page 52.

[124] Electronic Frontier Foundation. NSA Primary Sources. https://www.eff.org/de/nsa-spying/nsadocs. Cited on page 1.

[125] Electronic Frontier Foundation. The EFF SSL Observatory. https://www.eff.org/observatory. Cited on pages 28 and 36.

[126] Entrust. Entrust Bulletin on Certificates Issued with Weak 512-bit RSA Keys by Digicert Malaysia. Entrust Bulletin, Nov. 2011. http://www.entrust.net/advisories/malaysia.htm. Cited on page 46.

[127] European Network and Information Security Agency. Analysis of Operation Black Tulip: Certificate authorities lose authority. ENISA News, 05.12.2011. https://www.enisa.europa.eu/media/news-items/analysis-of-2018operation-black-tulip2019-certificate-authorities-lose-authority. Cited on page 44.

[128] D. Fisher. Another Dutch CA, KPN, Stops Issuing Certificates After Finding DDoS Tool On Server. threatpost, 04.11.2011. http://threatpost.com/another-dutch-ca-kpn-stops-issuing-certificates-after-finding-ddos-tool-server-110411/75855. Cited on page 47.

[129] FOX IT. Black Tulip - Report of the investigation into the DigiNotar Certificate Authority breach, 13.08.2012. http://www.rijksoverheid.nl/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update.html. Cited on page 44.

[130] genua and Technische Universität Darmstadt. Practical Hash-based Signatures. Project homepage, 2014. http://www.square-up.org/. Cited on page 144.

[131] GlobalSign. Security Incident Report. GlobalSign Press Release, 13.12.2011. http://dev.globalsign.co.uk/company/press/121311-security-incident-report.html. Cited on page 47.

[132] D. Goodin. New hack on Comodo reseller exposes private data – And then there were four. The Register - Security, 24.05.2011. http://www.theregister.co.uk/2011/05/24/comodo_reseller_hacked/. Cited on page 47.

[133] ICSI. The ICSI Certificate Notary. http://notary.icsi.berkeley.edu/. Cited on page 88.

[134] Internet Systems Consortium. ISC Domain Survey. http://www.isc.org/services/survey/. Cited on page 1.

[135] Internet World Stats. Internet Usage Statistics. The Internet Big Picture. http://www.internetworldstats.com/stats.htm. Cited on pages 1 and 35.

[136] joostbijl (pseudonym). RSA-512 Certificates abused in the wild. Fox IT Blog, 21.11.2011. http://blog.fox-it.com/2011/11/21/rsa-512-certificates-abused-in-the-wild/. Cited on page 46.

[137] J. Kersey. Stable and Beta Channel Updates - 10.0.648.151. Chrome Releases, 12.03.2011. http://googlechromereleases.blogspot.de/2011/03/stable-and-beta-channel-updates_17.html. Cited on page 43.

[138] A. Langley. Revocation checking and Chrome's CRL. ImperialViolet Blog, 05.02.2012. https://www.imperialviolet.org/2012/02/05/crlsets.html. Cited on page 39.

[139] A. Langley. Further improving digital certificate security. Google Online Security Blog, 07.12.2013. http://googleonlinesecurity.blogspot.com.es/2013/12/further-improving-digital-certificate.html. Cited on page 47.

[140] A. Langley. Maintaining digital certificate security. Google Online Security Blog, 08.07.2014. http://googleonlinesecurity.blogspot.de/2014/07/maintaining-digital-certificate-security.html. Cited on page 44.

[141] A. Langley. No, don't enable revocation checking. ImperialViolet Blog, 19.04.2014. https://www.imperialviolet.org/2014/04/19/revchecking.html. Cited on pages 39 and 54.

[142] A. Langley. Revocation doesn't work. ImperialViolet Blog, 18.03.2011. https://www.imperialviolet.org/2011/03/18/revocation.html. Cited on page 49.

[143] A. A. Lawati. Etisalat's BlackBerry update intercepts communication, says RIM. gulfnews.com, 21.07.2009. http://gulfnews.com/business/telecoms/etisalat-s-blackberry-update-intercepts-communication-says-rim-1.502062.. Cited on page 46.

[144] J. Leyden. Comodo admits 2 more resellers pwned in SSL cert hack – How deep does the rabbit hole go? The Register - Security, 30.03.2011. http://www.theregister.co.uk/2011/03/30/comodo_gate_latest/. Cited on page 43.

[145] M. Marlinspike. Convergence. http://convergence.io. Cited on pages 51 and 88.

[146] McAfee. SiteAdvisor. http://www.siteadvisor.com. Cited on page 33.

[147] J. Menn. Key Internet operator VeriSign hit by hackers. Reuters article, 02.02.2012. http://www.reuters.com/article/2012/02/02/us-hacking-verisign-idUSTRE8110Z820120202. Cited on page 47.

[148] Microsoft. Microsoft Root Certificate Program, 2009. http://technet.microsoft.com/en-us/library/cc751157.aspx. Cited on page 36.

[149] Microsoft. Fraudulent Digital Certificates Could Allow Spoofing. Microsoft Security Advisory 2641690, 10.11.2011. http://technet.microsoft.com/en-us/security/advisory/2641690. Cited on page 46.

[150] Microsoft. Fraudulent Digital Certificates Could Allow Spoofing. Microsoft Security Advisory 2798897, 03.01.2013. http://technet.microsoft.com/en-us/security/advisory/2798897. Cited on page 47.

[151] Microsoft. Improperly Issued Digital Certificates Could Allow Spoofing. Microsoft Security Advisory 2916652, 09.12.2013. http://technet.microsoft.com/en-us/security/advisory/2916652. Cited on page 47.

[152] Microsoft. Microsoft Releases Security Advisory 2524375. Microsoft Security Response Center, 23.03.2011. http://blogs.technet.com/b/msrc/archive/2011/03/23/microsoft-releases-security-advisory-2524375.aspx. Cited on page 43.

[153] Microsoft. MS01-017: Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard. Microsoft Support, 30.01.2001. https://support.microsoft.com/kb/293818. Cited on page 42.

[154] Microsoft. *Microsoft Application Architecture Guide, 2nd Edition*. Microsoft patterns & practices Developer Center, 2009. http://msdn.microsoft.com/en-us/library/ff650706.aspx. Cited on page 85.

[155] Mozilla Foundation. Network Security Services (NSS). https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS. Cited on pages 36 and 143.

[156] Mozilla Foundation. Mozilla CA Certificate Store - BuiltInCAs . https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/ Cited on page 36.

[157] Mozilla Foundation. Mozilla CA Certificate Policy - Version 2.2. https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/. Cited on page 36.

[158] Mozilla Foundation. CA:ImprovingRevocation. mozilla wiki, 01.10.2014. https://wiki.mozilla.org/CA:ImprovingRevocation. Cited on page 54.

[159] Mozilla Foundation. CA:RevocationPlan. mozilla wiki, 03.11.2014. https://wiki.mozilla.org/CA:RevocationPlan. Cited on pages 39 and 54.

[160] Mozilla Foundation. Simple Push. https://developer.mozilla.org/en-US/docs/Web/API/Simple_Push_API. Cited on page 116.

[161] Netcraft. Certificate revocation and the performance of OCSP, 18.04.2013. http://news.netcraft.com/archives/2013/04/16/certificate-revocation-and-the-performance-of-ocsp.html. Cited on page 54.

[162] Netcraft. Netcraft SSL Survey, 2013. http://www.netcraft.com/internet-data-mining/ssl-survey/. Cited on pages 54 and 58.

[163] E. Nigg. Full Disclosure. Join The Revolution! Personal Blog of Eddy Nigg, founder of StartCom, 03.01.2009. https://blog.startcom.org/?p=161. Cited on page 42.

[164] E. Nigg and R. Nigg. Critical Event Report. Start Commercial (StartCom) Limited – Incident Report, 20.12.2008. https://blog.startcom.org/wp-content/uploads/2009/01/ciritical-event-report-12-20-2008.pdf. Cited on page 42.

[165] J. Nightingale. Revoking Trust in DigiCert Sdn. Bhd Intermediate Certificate Authority. Mozilla Security Blog, 03.11.2011. https://blog.mozilla.org/security/2011/11/03/revoking-trust-in-digicert-sdn-bhd-intermediate-certificate-authority/. Cited on page 46.

[166] Norton. Safe Web. https://safeweb.norton.com. Cited on page 33.

[167] OpenSSL. Cryptography and SSL/TLS Toolkit. www.openssl.org. Cited on page 143.

[168] ORACLE. Java. Oracle Technology Network. http://www.oracle.com/technetwork/java/index.html. Cited on page 84.

[169] C. Osborne. Heartbleed: Over 300,000 servers still exposed. ZDNet, 23.06.2014. http://www.zdnet.com/heartbleed-over-300000-servers-still-exposed-7000030813. Cited on page 56.

[170] patrol.psyced.org. Certificate Patrol. http://patrol.psyced.org. Cited on page 49.

[171] pki.net.in.tum.de|crossbear.org. About crossbear. https://pki.net.in.tum.de/node/13. Cited on pages 51 and 88.

[172] B. Ray. BlackBerry update bursting with spyware – Official snooping suspected in UAE. The Register - Security, 14.07.2009. http://www.theregister.co.uk/2009/07/14/blackberry_snooping/. Cited on page 45.

[173] Research In Motion. RIM Customer Statement Regarding Etisalat / SS8 Software. RIM Customer Update, 17.07.2009. http://www.blackberrycool.com/wp-content/uploads/2009/07/blackberry-customer-statement-july-17-2009.pdf. Cited on page 45.

[174] M. J. Schwartz. How StartCom Foiled Comodohacker: 4 Lessons. DARKReading Attacks/Breaches, 09.08.2011. http://www.darkreading.com/attacks-and-breaches/how-startcom-foiled-comodohacker-4-lessons/d/d-id/1100043. Cited on page 44.

[175] SignatureCheck.org. Secure SSL/TLS Certificate Thumbprint Retrieval Service. https://www.signaturecheck.org. Cited on page 88.

[176] B. Smith. Bug 825022 - Deal with TURKTRUST mis-issued *.google.com certificate. Bugzilla@Mozilla, 27.12.2012. https://bugzilla.mozilla.org/show_bug.cgi?id=825022. Cited on page 47.

[177] SQLite Consortium. SQLite. http://www.sqlite.org. Cited on page 87.

[178] Statista. Prognose zum monatlichen Datenvolumen des privaten und geschäftlichen IP-Traffics weltweit von 2012 bis 2017 (in Exabyte). http://de.statista.com/statistik/daten/studie/266885/umfrage/prognose-zum-datenvolumen-des-privaten-und-geschaeftlichen-ip-traffics-weltweit/. Cited on page 1.

[179] Surety. The Power of Proof. http://www.surety.com. Cited on page 29.

[180] Telecooperation Group. CertainTrust SDK 1.0. Project homepage - Technische Universität Darmstadt. https://www.tk.informatik.tu-darmstadt.de/de/research/smart-security-and-trust. Cited on page 87.

[181] The Apache Software Foundation. Apache License, Version 2.0, 2004. http://www.apache.org/licenses/LICENSE-2.0. Cited on page 84.

[182] Trustwave. Defective By Design? - Certificate Revocation Behavior In Modern Browsers. SpiderLabs Blog, 04.04.2011. http://blog.spiderlabs.com/2011/04/certificate-revocation-behavior-in-modern-browsers.html. Cited on page 39.

[183] Trustwave. Clarifying The Trustwave CA Policy Update. SpiderLabs Blog, 04.02.2012. http://blog.spiderlabs.com/2012/02/clarifying-the-trustwave-ca-policy-update.html. Cited on page 46.

[184] P. Turner, P. Turner, and E. Barker. Preparing for and Responding to Certification Authority Compromise and Fraudulent Certificate Issuance. ITL Bulletin. July 2012. http://csrc.nist.gov/publications/nistbul/july-2012_itl-bulletin.pdf. Cited on page 40.

[185] D. Veditz. Bug 642395 - Deal with bogus certs issued by Comodo partner. Bugzilla@Mozilla, 17.03.2011. https://bugzilla.mozilla.org/show_bug.cgi?id=642395. Cited on page 43.

[186] Veracode. BlackBerry Spyware Dissected. Veracode Blog, 15.07.2009. http://blog.veracode.com/2009/07/blackberry-spyware-dissected/. Cited on page 45.

[187] Verisign. Quarterly Report pursuant to Section 13 or 15(d) of the Securities Exchange Act of 1934. Verisign Inc/CA Sec filing Form 10-Q, 2011. https://investor.verisign.com/secfiling.cfm?filingID=1193125-11-285850&CIK=1014473#D219781D10Q_HTM_TOC. Cited on page 47.

[188] Wiki - The Hacker's Choice. SSL/TLS in a Post-Prism Era. https://wiki.thc.org/ssl. Cited on page 42.

[189] K. Wilson. Revoking Trust in one ANSSI Certificate. Mozilla Security Blog, 09.12.2013. https://blog.mozilla.org/security/2013/12/09/revoking-trust-in-one-anssi-certificate/. Cited on page 47.

[190] WOT - Web Of Trust. Know which websites to trust. https://www.mywot.com. Cited on pages 33 and 75.

Appendix

Details on data sets extracted from browsing histories

Trust View	Number of hosts	Number of hosts assigned to CAs with trustworthiness w/o RS			with RS		
		full	med	min	full	med	min
1	2	0	0	0	0	0	1
2	4	0	0	0	0	1	2
3	5	0	0	1	0	2	2
4	5	0	0	0	0	1	3
5	10	0	3	1	3	1	4
6	15	0	6	4	3	7	5
7	21	0	11	3	12	4	4
8	22	7	5	2	14	5	3
9	26	0	19	3	19	3	2
10	44	4	25	5	26	14	4
11	46	7	23	7	37	5	2
12	47	7	18	8	26	11	8
13	48	0	22	13	24	14	10
14	48	11	24	7	33	13	2
15	51	9	26	6	35	6	8
16	55	9	21	13	30	17	8
17	63	19	19	13	43	13	5
18	68	18	25	16	48	16	4
19	83	55	10	3	66	9	7
20	84	42	23	8	65	11	7
21	90	43	18	15	58	15	17
22	94	38	30	12	66	21	7
23	95	50	22	14	76	8	11
24	101	44	30	15	78	14	9
25	102	45	30	17	72	22	6
26	105	44	38	19	84	15	5
27	107	31	45	18	78	20	9
28	110	62	21	14	85	11	14

29	134	53	48	20	101	22	9
30	144	99	22	13	118	19	7
31	151	86	37	16	113	30	8
32	152	88	40	13	131	12	7
33	157	78	47	15	110	34	13
34	166	96	39	21	131	18	15
35	176	125	25	13	146	15	14
36	178	93	62	13	119	46	11
37	180	72	75	21	129	37	6
38	181	93	68	13	156	18	7
39	202	123	48	14	158	26	8
40	203	137	46	7	173	15	11
41	218	134	57	20	176	25	13
42	240	160	49	18	191	34	10
43	255	168	58	15	208	30	13
44	278	213	37	15	238	26	7
45	281	188	71	19	249	24	8
46	330	245	54	19	275	39	8
47	340	236	77	15	291	33	10
48	341	233	78	21	262	60	13
49	341	233	71	19	273	53	9
50	347	278	38	19	299	33	10
51	351	256	64	23	283	49	13
52	371	267	80	15	329	32	8
53	388	275	81	19	328	37	12
54	400	289	84	20	326	64	7
55	419	337	51	19	355	46	11
56	463	392	50	15	424	30	5
57	466	384	60	14	425	32	5
58	475	395	55	13	438	29	7
59	506	425	61	15	455	41	8
60	532	447	54	22	483	37	8
61	540	453	59	21	491	37	8
62	604	517	65	13	543	47	10
63	639	535	76	17	575	49	5
64	1013	903	77	20	912	82	8

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

Trust View	Number of CAs	Number of host signing CAs	Number of CAs with trustworthiness w/o RS			with RS		
			full	med	min	full	med	min
1	6	2	0	0	0	0	0	1
2	7	3	0	0	0	0	1	1
3	11	5	0	0	1	0	2	2
4	10	4	0	0	0	0	1	2
5	16	7	0	1	1	1	1	3
6	17	10	0	2	3	1	4	5
7	25	13	0	3	3	4	4	4
8	23	11	1	1	2	4	4	3
9	17	10	0	4	2	4	2	2
10	38	19	1	5	4	8	7	4
11	37	20	1	6	6	13	4	2
12	41	24	1	5	5	7	9	6
13	44	25	0	5	9	8	8	9
14	31	18	1	6	5	8	8	2
15	39	22	2	6	4	8	4	8
16	40	26	2	5	9	8	11	7
17	49	27	2	6	9	12	9	4
18	48	29	3	7	12	14	11	4
19	46	27	8	3	3	12	7	7
20	49	26	5	5	6	10	9	6
21	52	33	6	5	10	12	7	14
22	57	32	5	8	9	15	12	5
23	54	32	6	7	10	16	5	11
24	63	37	5	9	12	17	12	8
25	59	34	6	7	12	15	12	5
26	49	30	3	10	13	15	9	5
27	55	34	4	10	11	15	12	7
28	48	30	6	5	9	12	7	11
29	75	48	6	16	15	23	16	7
30	58	38	11	8	11	17	14	7
31	66	42	8	11	13	16	18	8
32	71	44	11	13	11	26	11	6
33	72	48	8	17	10	19	18	11
34	69	44	9	13	14	19	12	11
35	64	39	12	6	10	17	10	11
36	80	59	10	30	12	18	31	9
37	91	61	7	27	17	22	27	5
38	65	44	8	18	12	24	13	7
39	79	50	11	15	11	20	16	7
40	69	43	14	11	7	23	7	10
41	72	52	12	20	15	21	18	11
42	88	59	14	19	15	23	22	9
43	80	52	13	16	13	21	18	9

44	93	60	22	16	11	29	19	6
45	86	57	19	21	14	36	16	5
46	94	66	22	18	16	30	21	7
47	89	59	15	21	13	27	19	7
48	97	73	17	32	15	24	33	10
49	114	79	20	29	16	31	35	7
50	83	56	21	10	15	25	18	8
51	112	80	22	33	18	31	34	10
52	90	62	18	25	12	32	22	6
53	108	73	21	28	13	32	23	9
54	113	83	23	40	14	33	42	5
55	105	71	26	22	14	32	25	8
56	79	54	25	18	8	34	15	3
57	95	65	26	19	12	36	20	5
58	104	68	28	20	11	39	22	6
59	103	70	29	25	11	36	26	6
60	111	74	30	19	17	40	24	7
61	109	74	31	21	15	41	22	7
62	103	76	29	29	11	35	30	7
63	132	91	33	32	15	46	31	4
64	138	98	45	33	13	47	41	4

Table 2: Numbers of CAs and host signing CAs and their respective trustworthiness.

Wissenschaftlicher Werdegang

Mai 2010 - heute Wissenschaftlicher Mitarbeiter in der Arbeitsgruppe von Professor Johannes Buchmann, Fachbereich Informatik, Fachgebiet Theoretische Informatik – Kryptographie und Computeralgebra an der Technischen Universität Darmstadt.

Oktober 2004 - Dezember 2009 Studium der Wirtschaftsinformatik (Diplom) an der Technischen Universität Darmstadt.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbständig verfasst habe.

Darmstadt, März 2015 _____