Master Game and Media Technology

Improving PKI

Solution analysis in case of CA compromisation

Samira Zaker Soltani

January 2013

Universiteit Utrecht

Utrecht University
Faculty Computer Science

Deloitte.

Deloitte Nederland
Deloitte Risk Services

Supervisors:
Gerard Tel - Univeristy Utrecht
Henk Marsman - Deloitte Nederland

To my mother, for she is the reason.

Abstract

Creating a secure connection on the Internet is made possible through the usage of certificates, binding an entity to its public key. These certificates can be issued by any of the Certificate Authorities (CA), where each CA has the same privileges. During the last year, we have seen many CA compromises, resulting into the issuance of fraudulent certificates. Fraudulent certificates can be used, in combination with the man-in-the-middle attack, to eavesdrop the communications of Internet users.

This research focuses on solutions that can remove or limit the impact of a CA compromisation and provides a description and analysis of each solution. The solutions have been chosen through interviews and literature. Among the discussed solutions are Public Key Pinning, Sovereign Keys, Certificate Transparency, Perspectives & Convergence, DANE, and MCS.

In order to identify each solution's advantages and disadvantages, we have created a metric of aspects. The aspects have been categorized into security, usability, and costs. The focus of this research has been on security, since that is the aspect in Public Key Infrastructure we are trying to solve.

The results indicate that Certificate Transparency and DANE are the most promising solutions for limiting the risks of a compromised CA. Further research will be needed to complete each solution, since both solutions are not yet ready for deployment.

Preface

This document presents my master thesis for Game and Media Technology at the Utrecht University. The research and work was done between July 2012 and January 2013 at the Security & Privacy team of Deloitte Nederland in Amstelveen. My supervisors were Henk Marsman from Deloitte Nederland and Gerard Tel from Utrecht University.

Acknowledgment

I am grateful to Gerard Tel for his guidance during this master thesis, the inspiring discussions, encouragements, and for giving me the freedom to take this project into the direction that I found most interesting. I would also like to express my sincere appreciations to Henk Marsman for his extensive vision, great wisdom, and for helping me to better structure my master thesis. I would also like to thank the interviewees, who helped me to develop the indispensable information that was needed for this research. Without the interviewees, this master thesis would not have been the same. Finally, I would like to thank my colleges for the enlightening discussions, remarks, and for the overall atmosphere that made my stay at Deloitte very pleasant and enjoyable.

Samira Zaker Soltani

Contents

1. Introduction

Internet security has become more and more important since the Internet started changing our world and E-commerce became a part of our business and personal life. In the late 90's, Public Key Infrastructures (PKIs) were widely recognized as an essential ingredient to provide secure electronic communications and transactions in open environments [FB00, FW98].

The most important goal of PKI is to secure the communication between parties in an insecure public network, such as the Internet, and make sure the parties communicate with the parties they think they are communicating with. In the last year, however, PKI has shown not to be as bullet proof as we hoped it would be.

PKI works through the use of certificates, which can be thought of as digital passports. These digital passports are issued by Certificate Authorities (CAs) to various parties. A certificate guarantees a party's identity, allowing the party to identify itself towards users on the Internet. Users can validate the certificate authenticity by checking if the certificate has been issued by a trusted CA. Once the certificate has been validated, the user can start communicating with the party corresponding to the certificate. If a CA is not trusted by the user, they can refuse to start a communication with the party presenting the certificate. To not burden users with the choice between the approximately 600 CA organizations in the world, this choice is performed by the user's browser and operating system software. Most users will never change these settings and keep trusting the default set of CAs [Lan11f].

PKI has a design weakness though, granting all CAs to issue certificates for any domain, which will be accepted by users trusting the CA. Compromised, but still trusted CAs, can issue certificates for any domain and use these certificates to deceive users into thinking the certificates are from the righteous party. This means that when a by browsers trusted CA is compromised, many users will trust the fraudulently issued certificates.

This weakness was shown in practice when DigiNotar, a trusted CA, was compromised and used to issue fraudulent certificates for domains such as Google and Yahoo! in 2011. Using these certificates, the attackers could, for example, pretend to be a legitimate website, where visitors would submit their credentials such as username and password, as the presented certificates seemed valid.

Although the chance of a CA compromisation was thought to be very little, after some trusted CAs were compromised in 2011, with the most important incident being the DigiNotar compromisation in July 2011 [Dig12], it became clear that this

was not the case. The incidents revealed some interesting problems in PKI, regarding trust anchor management, the dependency of the complete Internet PKI on the weakest trusted CA, the lack of preparation at companies when a CA compromisation occurs and many more weaknesses.

Of course, if the compromised CA would be detected immediately, there would be little to no problem, depending on the time frame. However, if we take DigiNotar as an example, we can see it took a couple of months before the trust in DigiNotar was revoked by all browser and operating system vendors, giving the hacker enough time to use the fraudulently issued certificates. The only thing we know for sure, regarding the DigiNotar incident, is that the OCSP requests were mostly from Iran [Pri11]. OCSP is an Internet protocol used to obtain the status of a certificate. When users want to know the status of a certificate, they can send an OCSP request to the corresponding CA. The CA will then answer whether the certificate is still valid or not. In the case of DigiNotar, they were able to revoke most of the fraudulently issued certificates. Some of the certificates, however, could not be revoked, because DigiNotar simply was not able to identify which certificates had been issued fraudulently, see Sec. 1.1.1.

Using the information from the OCSP request, we know that the certificates were used. However, we can only speculate the hacker used the certificates for attacks to steal from people, eavesdrop email conversations or obtained other personal data.

Let us summarize the problem of the Internet PKI in one sentence: Until compromised CAs are not detected and distrusted by browsers and operating systems, we can be sure that we can not be sure about the safety of our conversations.

An at most interesting problem, leaving us with a straight forward goal for our research: how can we improve the Public Key Infrastructure used for Internet.

Although this was first my goal, the creation of an improved PKI was too ambitous in the timeframe of a master thesis. Since various parties have already tried this in the last year, the new goal became to *analyze the proposals for improving the Public Key Infrastructure, either by limiting or eliminating the security risks, which occur since the period a Certificate Authority is compromised until the recovery of the PKI, regardless whether the compromisation has been detected or not.*

The many Certificate Authority compromisations that led to understanding the weaknesses in the current Internet PKI, being also the motivation of this research, are explained in Sec. 1.1. DigiNotar, the most important Certificate Authority compromisation of 2011, has been explained separately in Sec. 1.1.1. In Sec. 1.2 the goal, research questions, and restrictions of this research are specified. The approach for the research is announced in Sec. 1.3.

1.1. Motivation

Previously, we have explained that the Internet Public Key Infrastructure is designed such that if only one of the trusted Certificate Authorities is compromised, all supposedly safe communications can not be considered secure. Until the compromised CA is distrusted or its false certificates are revoked, the communications will remain unsafe.

The obvious problem here is that users are not safe in the time between a CA compromisation and until the time the actions against it have been completed. This time period was less than an hour in the case of Comodo in March 2011 [Com11a], but more than 8 weeks in the case of DigiNotar [Lan12]. Both Comodo and DigiNotar were trusted by all browsers and had issued fraudulent certificates.

Both of the cases were very serious, since the hacker could issue a certificate for the most widely used web browsers and email clients, which were trusted by Internet Explorer (IE), Firefox, Chrome, Safari and Opera [Lan12]. The reason the compromisation at DigiNotar resulted to their bankruptcy, was because DigiNotar could not figure out which certificates were fraudulent and was therefore not able to confirm whether they had revoked all fraudulently issued certificates.

More recent cases of 2012 compromisations were the detection of fraudulently code signed software. Code signing is another application of PKI, where the signature of the trusted CA guarantees the author of the code. This is for example used for Microsoft updates, assuring the updates for windows are made by Microsoft and not a malicious party. However, also in this part of PKI, CA compromisations are possible. Both of the following examples happened in 2012. The first was the Flame virus, signed by Microsoft Root Authority in June [Nes12], and the second were the two malicious utilities that appeared to be digitally signed by a valid Adobe code signing certificate [Ark12].

All of the compromised CAs, except for DigiNotar, were able to find the leak, solve the problem and discussed the compromise publicly. Comodo, for example, was able to revoke the fraudulently issued certificates within the hour and solved the problem with their compromised reseller. Although Microsoft and Adobe were not this fast, taking Adobe more than 2.5 months until the compromisation was discovered, they were fairly open about the subject. The unfortunate faith of DigiNotar going bankrupt was not only because their infrastructure was very insecure, but also due to the fact nobody trusted them anymore. DigiNotar knew they were hacked, but chose to keep this information from the world, in order to protect their own image. We will discuss the case of DigiNotar in more detail in Sec. 1.1.1.

At 5th of September 2011, the hacker of both Comodo and DigiNotar, calling himself the Comodo Hacker, posted on Pastebin [Com11c] that he had access to 4 other big Certificate Authorities: *"You know, I have access to 4 more so HIGH profile CA's, which I can issue certs from them too which I will, I won't name them"*. At 6th of

September the Comodo Hacker posted on Pastebin [Com11b]: *"I still have access to 4 more CA's, I just named one and I re-name it: GlobalSign".*

"Everything that can go wrong, will go wrong" Murphy's Law.

Although we can not be sure he is telling the truth, the Comodo Hacker did prove he can hack into highly secured Certificate Authorities [Gra11], which is a huge problem. In the investigation paper after the DigiNotar incident published by the Dutch Safety Board [Dig12], they wrote that there is no such thing as being 100% secure, and that this also holds in Digital security. Therefore, we must always take into account that any system we build, will be hacked and keep searching for solutions to improve our weaknesses.

The weakness in PKI, where the security of the complete network is dependent on the weakest trusted CA, should therefore be solved or minimized. In this research we try to analyze existing proposals and help in the quest towards a better PKI.

With the forthcoming of Internet in the last decades, not only security in e-commerce has taken an important place, but also the rapidly growing online gaming industry. Research has suggested that cheating is a major security concern for online games, where only a minority of cheaters can potentially ruin the game for all players and destroy a game's success. An additional research goal we therefore want to pursue, is researching this major security issue in online games, and possibly solving some of the issues by using PKI.

1.1.1. DigiNotar

DigiNotar was a Certificate Authority, initiated in 1997 by the Koninklijke Notariële Beroepsorganisatie (KNB), offering technical services and issuing digital certificates to notaries. Certificates issued included the default SSL certificates, Qualified Certificates and 'PKIoverheid' (Government accredited) certificates. Certificates for the Dutch government's PKIoverheid were issued by DigiNotar in 2004. An important example of certificates issued by DigiNotar was for the authentication infrastructure DigiD, which is an identity management platform used to verify the identity of Dutch citizens on the Internet.

In 2006, DigiNotar obtained the Webtrust license which is given to reliable websites concerning good care of information and personal information of visitors. A yearly audit was held to allow DigiNotar keep this license. DigiNotar also obtained the TTP.NL statement, adding another yearly audit to the company. DigiNotar was sold to Vasco Data Security Internation Inc. on 10 January 2011.

In the period starting from June 2011 until the bankruptcy of DigiNotar, many events have occured. Multiple mistakes were involved, not only by DigiNotar, but also the companies doing the audits and giving permissions to issue certificates. However, it seems that nobody wants to point a finger at anything or anyone in particular [Dig12].

Fox-IT was asked by the Dutch government to do an investigation on the DigiNotar incident and published the report called 'Operation Black Tulip' [Pri11]. The technical report shows that the attacks were carried out in multiple phases in a period of a few weeks. The first traces of hacker activities started on June 17th, while the first succeeded rogue certificate was issued at 1 July.

After DigiNotar was aware of the security leak, they immediatly placed the suspected certificates on their Certificate Revocation List, meaning the certificate will become invalid. Further investigation, however, revealed that other malicious certificates were issued. However, DigiNotar was not able to trace the id of these certificates, meaning that DigiNotar was not able to revoke these certificates. Therefore the decision was made to distrust the complete root of DigiNotar. Although, this was the best decision the Dutch Government could take at the time, DigiNotar was not removed from the trusted root CAs list immediately due to the problems it may have caused to the Dutch governments infrastructure. This of course extended the time period where we can say for sure that the world was not completely safe. Even now, users using outdated software or browsers, where DigiNotar certificates are still trusted, can be unsafe from the same attacks.

For those interested, a time line of the events can be found in the Fox-IT report [Pri11]. In the next section, the main reasons to revoke the trust in DigiNotar's root CA are mentioned. Soon after the revocation of DigiNotar's root CA from the trusted root CAs, the company was announced bankrupt.

Main reasons for revoking DigiNotar's trust

- Companies who were supposed to do an audit on DigiNotar, making sure DigiNotar still met the requirements in order to extend DigiNotar's licenses, did not perform the audit in practice, but only on paper [Dig12].
- DigiNotar's program that performed a check on the issued certificates against the requested certificates was not working for unknown reasons and for an unknown period of time. The program was recovered on the 19th of July, 19 days after the first rogue certificate was issued by the hacker. This is remarkable since DigiNotar had noticed some signals of the approach of a possible attack. They had received warnings, through an email, about intensified hack activities [Dig12] by Logius, the digital government service of the Netherlands Ministry of the Interior and Kingdom Relations. The only precaution DigiNotar took at that time, however, was blocking some IP-addresses and notifying their employees about the situation.
- The most critical servers, investigated by Fox-IT, contained malware. This could have been easily detected if the servers had an anti-virus software. None of the investigated servers had an anti-virus [Pri11].
- A few servers used by the hacker to obtain more access in DigiNotar's network, ran on outdated software [Dig12].

- In the Fox-IT report [Pri11] they mentioned that the CA-servers, although physically very securely placed, were accessible over the network from the management LAN. All CA-servers were members of one Windows domain, which made it possible to access them all using one user/password combination.
- An addition to the previous reason, the password with administrative rights was not very strong and could easily be brute-forced [Pri11]. The administrative account had access to the software issuing certificates [Dig12]. The secret keys needed to create the Certificate Revocation List, were available in a protected part of the network and could be accessed and used by the hacker.
- When DigiNotar found out about the fraudulently issued certificates on 19th of July, they decided to keep it from the world, even after consultation with Vasco. This lowered their credibility when the exploit was discovered.

1.2. Goal

Agreeing with Murphy's Law, the goal of this research is not to find solutions that prevent Certificate Authorities from being compromised, but to identify solutions to limit or eliminate the security risks during the compromisation of a CA, whether this is already detected or not. The many Certificate Authority compromisations during the last year have been the provocation that was needed to show the problems with the current Internet PKI and it has given a reason to start the search for a solution. The abuse of a single CA is sufficient to generate valid certificates on behalf of any entity, trick clients into accepting this fraudulent but valid certificate and perform a successful attack, such as the man-in-the-middle-attack. Therefore, we define our goal as such:

The goal is analyzing proposals to improve the Public Key Infrastructure, either by limiting or eliminating the security risks, which occur since the period a Certificate Authority is compromised until the recovery of the PKI, regardless whether the compromisation has been detected or not.

1.2.1. Research questions

The research questions that will help to find a solution to the problem are:

- How can we improve the current Public Key Infrastructure to limit or eliminate the security risks?
  - How does PKI work?
  - How is PKI used in companies and in the game industry?
  - What are the weaknesses and vulnerabilities of the current PKI?

- What solutions are there to limit or eliminate these weaknesses?
- Which solution is best suited?

1.2.2. Research restrictions

There are many different Public Key Infrastructures networks, and in most cases, PKI is customized to satisfy the requirements for that network. The Internet is the largest PKI network. The focus of this thesis is to analyse solutions for Internet PKI. Other PKIs are considered to be outside the scope of this research. Possibly, the results or some aspects of it, will also help other PKIs.

It is important to understand this research is aiming to limit or eliminate the risks resulting from a compromised CA, as was the case with DigiNotar. The research will not aim to improve audits, CA's processes, norms, and regulations.

1.3. Approach

To get familiar with the problem we have first started with a literature study about PKI. During this period, we decided that it would be a good practice to hear the opinions of professionals and companies on the subject through interviews. These interviews have provided us with insight and information that would have been harder to find in literature, and could only be obtained by experience. After the literature study and interviews, we were able to create an overview of the workings and weaknesses of PKI, see chapter 2.

Using the information from the interview and the literature study, we then started searching for solutions to limit or decrease the risks of a CA compromisation. To allow analyzing the multiple solutions we had found during the first research period, we created a metric for the solution's aspects, which is defined in Sec. 1.3.1. The aspects overviews made it easier to analyze and determine the solutions that would fits the goal of this research.

For the research about security in computer games, possibly involving certificates, we have also tried to get interviews from game companies with an online game, since the literature about online game security is limited. Although the biggest game companies did not agree on an interview, we have succeeded to interview three companies in the gaming industry and a game hacker. The interviews and the literature helped us to document the security issued in online games and their possible counter measurements, see chapter 4.

1.3.1. Solution aspects

After the realization that there are multiple solutions available to improve the current PKI, we started searching for a standard criterium to categorize the differ-

ent solutions. For this, we found literature about balancing usability against costs [Kar94], where usability means that a product will be easy to learn, efficient to use, and satisfying for users. However, there was no literature about criteria for designing new protocols that also include security, and have therefore created my own matrix to categorize the solutions. The solutions will be categorized by security, usability, and costs.

Since PKI's reason for existence and the reason to improve it, is to add more security to user's communications, *security* will be the main categorie of the solutions. For this categorie, four aspects are defined, covering the security area: Risks, Improvements, Sustainability, and Juridical. Each aspect of this category will be explained later in this section.

Usability is the second category, since it is more important for this research to find a solution that will be usable, including users, domain holders, server administrators, etc. As the solutions will be explained and analyzed in chapter 5, the reader will understand that most solutions are still in their design phase. Therefore, it is difficult to specify explicit guidelines for the solutions, which will be needed later on, when one or multiple solutions have shown to have the capacity to improve PKI.

According to research in case studies for balancing usability and costs, it has shown that increasing usability during the design phase will cost many times less than during the product release [Kar94]. Although this is a different product, it is clear that the same holds for these solutions. Even the smallest change to fix some design mistake, after the design phase, will cost more. Therefore, for our matrix, usability is more important than costs. In order to cover usability, we have devided this category into the following aspects: Efficiency, Control, Availability, and Limits.

The new solution will have some transition time, after which it will become fully operational by either improving or replacing PKI. Transition costs are therefore the first aspect of the cost category, where maintaining the components of the solution is the second aspect. It is not possible to make changes without having to explain them to any of the many different users of PKI. PKI has many different users, which include for example Internet users, domain owners, system administrators, and programmers at browser vendors. After any change, educational costs will be needed, dependent on which part of PKI the changes will be carried through. An additional aspect of the cost category is the losses and profits, summarizing which parties will experience losses or profits if the solution would be implemented.

These three categories will be used as a guideline to identify whether a solution will be sufficient to limit the risks of a compromised CA and will be feasible for implementation. Since most solutions are still in their first phases of development, mostly design, it is not possible to compare them with exact guidelines, and are therefore used to help identifying whether a solution will have the capacity to improve Internet PKI. The matrix, however, can be used for further research, but will need to be extended towards specific guidelines per aspect.

Security:

Risks: It is necessary to know what the (new) risks of this new solution will be in normal use, its single point of failures, and the risks when a CA is compromised. Risks under normal usage of PKI can for example be MITM attacks, privacy issues, failures of the solutions resulting into Denial of Service or false certifiate warnings.

Improvements: The solution we seek needs to improve the level of security in comparison of the current model. The best possible solution would be able to remove all of the PKI weaknesses that are discussed in Sec. 2.6.

Sustainability: As technology is always improving and changing, the new solution should be able to endure through renewals of thechniques. It is also necessary to identify the solution's dependencies. Needless to say that a perfect solution would have no dependencies and will sustain or be able to adjust itself through renewal of techniques.

Juridical: The parties responsible for critical parts of the solution must be identified. It is also important to identify whether these parties can be held responsible when something goes wrong and whether they can be influenced by for example governments to perform illegal actions. It is preferred to be dependent on official parties and have the ability to held them responsible when something goes wrong. It is also preferred to have a decentralized trust, where a large company or government, will not be able to gain control over (parts of) PKI.

Usability:

Efficiency: A studie in web usability has also shown that web pages should be plain, simple, and fast, since users do not have the tolerance to wait or learn about web pages [NN00]. The new solution should therefore not cause significant latency to the system and be easy to use.

Control: The amount of control over the trust by users must be identified and whether it is possible for users to visit a domain anyway, after its certificate has been marked as invalid. Since most users will never change the default settings [Lan11f], less control for users will be preferred, while the solution assures their security with the default options. However, it is preferred to have the option to distrust an entity for the remaining users that do want to be in control of their trust or when for example the same situation arises that distrusting a compromised CA is delayed, as was the case with DigiNotar [Dig12].
Similarly, it is preferred to disallow users to proceed to websites with invalid certificates, since research suggests that web users often make bad security decisions by ignoring warning messages [ECH08]. This does require the solution to work as desired and does not allow false positives.

Availability: Identify circumstances when the system will not be available and whether

the available resources will be used by the new solution. Preferred is ofcourse when changes do not break older software and hardware

Limits: New solutions may have additional limitations. An example would be if the solution was not to be used by large companies by allowing unofficial parties to take responsibility over important components of the solution. Another example would be if the solution was only available for web pages.

Costs:

Transition costs: This is the cost needed to change the infrastructure or software in order to be able to use the new solution. Preferred is when the solution only requires changes to webbrowsers or CAs, since there are only five major browser vendors and around 600 CAs [PE11]. Changes to the tens of millions of web hosts or several billion users is less preferred .

Maintenance costs: This aspect covers the cost needed to maintain the solution's components. Because most solutions are not yet in this stadium, this aspect will be very difficult to figure out.

Educational costs: This is the cost needed to educate personnel and users to be able to work with the new solution. Although educating employees from webbrowsers and CAs are significantly less in numbers, than educating web administrators for example, this depends very much on the amount of needed education.

Losses and profits: This aspect will give an insight of which parties are expected to experience losses or profits if this solution would be used. The solution will definitely have some costs and these costs must be paid by some party. Beside preferring lower costs, it will probably increase the acceptance of the solution when losses would be for large organizations such as browser vendors and CAs, instead of every domain owner.

Table 1.1.: This is a summary of the various aspects, grouped by type, mentioned in the previous paragraph of the soultion analyts.

Security	Usability	Costs
Risks	**Efficiency**	**Transition costs**
Improvements	**Control**	**Maintanance costs**
Sustainability	**Availability**	**Educational costs**
Juridical	**Limits**	**Losses and profits**

2. Public Key Infrastructure

2.1. Overview

In this chapter, the working of Public Key Infrastructure, the idea behind it and its weaknesses are explained, which will help readers to understand further chapters. Readers already familiar with PKI are encouraged to read only the last section of this chapter, where PKI weaknesses are discussed.

The chapter first begins in Sec. 2.2 with explaining the tale of the cryptographic heart that is essential for PKI, whereupon the overview of PKI is given in Sec. 2.3. In sections Sec. 2.4 and Sec. 2.5, the two important parts of PKI, Certificates and Certificate Authorities, are explained in more detail. We close this chapter with Sec. 2.6, discussing the weaknesses in the current PKI.

2.2. Cryptography

Cryptography is the science of keeping secrets secret [DK01]. It is the practice and study of techniques to allow confidentiality and secure communication between parties in the presence of adversaries [Riv90].

In cryptography and what follows, the sender of the message will be referred to as Alice, the receiver as Bob, and the adversary as Eve. The by humans readable message is referred to as plain text and the unreadable message is called cipher text. The process of transforming a plain text to cipher text is called encryption. Decryption is the reverse process, where cipher text is transformed to plain text using a secret key.

Encryption of data is possible by using either secret (symmetric) or public key (asymmetric) cryptography [Wei01]. In Sec. 2.2.1 a brief overview of symmetric key cryptography is given and in Sec. 2.2.2 the public key cryptography is explained.

Cryptography is also used to provide solutions for other problems than confidentiality: data integrity, authentication and non-repudiation [DK01].

Data integrity means that the receiver of a message should be able to check whether the message was modified during transmission, either accidentally or deliberately. Similarly, when messages are written on special paper, the paper provides a certain

security against manipulation. Eve should not be able to substitute a false message for the original message, nor parts of it.

Authentication means the receiver of a message should be able to verify the origin of the message. Bob must be able to verify the message is from Alice and not Eve.

Non-repudiation means the sender should not be able to later deny that the sent a message.

Handwritten signatures are intended to guarantee authentication and non-repudiation. Using public key cryptography digital signatures can be created and is explained in Sec. 2.2.2.

2.2.1. Symmetric key cryptography

In symmetric cryptography, both communication parties use the same key for encryption and decryption. Symmetric key encryption provides secrecy when two parties communicate. It is important that this key is kept secret and only known to Alice and Bob, otherwise an adversary, Eve, will be able to decrypt and read the private messages between Alice and Bob. Important examples of symmetric key encryption schemes are 3DES and AES.

An important problem when using symmetric key cryptography, called the key exchange problem, is that Alice and Bob must agree on a secret key in a secure and efficient way that will allow them to communicate securely [DK01]. In many application, such as the Internet, where it can occur that any two parties communicate with each other for the first time, symmetric cryptographic techniques are not desired, due to the key exchange problem [Bra00].

There was no solution to the key exchange problem until the revolutionary concept of public key cryptography in 1976 by Diffie and Hellman in [DH76]. This concept will be explained in Sec. 2.2.2.

Symmetric key encryption algorithms have the fastest implementations in hardware and software and can be used for large amount of data. The public key cryptography, which is more complex and less efficient for large amount of data, is often used to provide a secret key which can be used for further communication. Thus, symmetric key and public key cryptography complement each other to provide practical cryptsystems.

2.2.2. Public key cryptography

In this section we will give a brief overview of one of the most important cryptography used in PKI. The public key cryptography explained here is mostly attributed

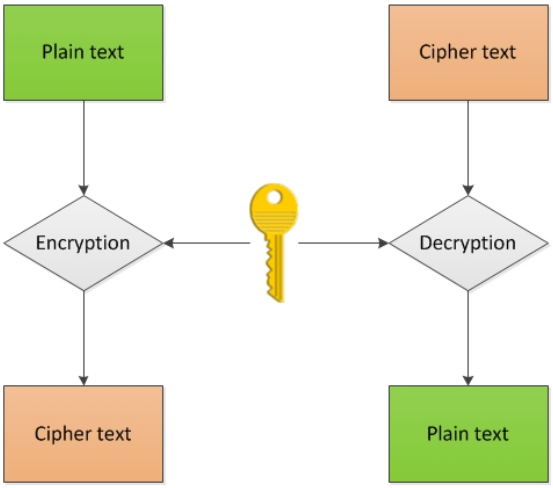

Figure 2.1.: Symmetric key cryptography

to Diffie and Hellman [DH76], who were one of the first to give a practical example of key exchange in an insecure network, and Rivest, Shamir and Adleman [RSA78], who followed this and gave another implementation of public key cryptography.

Public key cryptography is an important and widely used technology. It is for instance used in Internet standards such as Transport Layer Security and Pretty Good Privacy.

In a public key encryption scheme, the communication partners do not share a secret key. In public key cryptography a person's key is separated into two parts: the public key for encryption available to everyone and a private key for decryption which is kept secret by the owner.

Suppose Alice wants to send a message to Bob using public key cryptography. Using Bob's public key, Alice can encrypt messages for Bob. After receiving the messages, Bob can decrypt them using his private key.

Everyone can easily encrypt a plain text using the public key, but only the one with the private key can decrypt messages, see Fig. 2.2. Thus, knowing a person's public key does not allow the decryption of an encrypted message to that person by anyone that does not posses the private key. Mathematically speaking, public key encryption is a so-called one-way function with a trapdoor. We speak of a one-way function when the encryption of a plain text to cipher text is computable by an efficient algorithm, but practically impossible to deduce the plain text from the cipher text, without knowing the trapdoor information. The trapdoor information

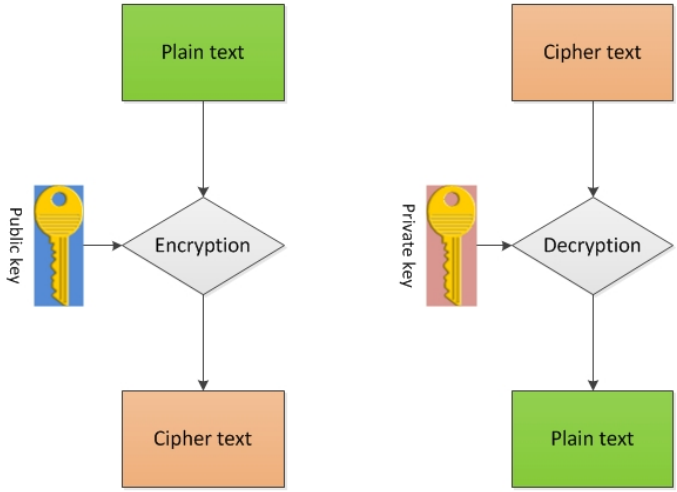

Figure 2.2.: Public key cryptography

in public key cryptography is the private key. One-way functions with this property are called trapdoor functions [DH76].

The private key should be kept secret and stay only known to the owner, the creator of the public and private key pair. According to this, Alice and Bob can communicate with each other without any prearranged secret keys. Using public key cryptography, Alice and Bob only need to fetch the publicly available public key of each other.

In 1976, Diffie and Hellman described, in addition to their key agreement method, how digital signatures would work and proposed, as an open question, the search for such a function.

A digital signature is another means to ensure integrity, authenticity, and non-repudiation. A digital signature, is an electronic signature that can be used to authenticate and verify the identity of the sender of a message (non-repudiate) as well as the integrity of the data. A digital signature can be used with any kind of message, encrypted or not, to show the the sender's identity to the receiver of the message. The digital signature also makes sure that modifications to the message, after the signature, will be detected.

A digital signature is derived by applying a mathematical function, to compute the hash value of an electronic message, and then encrypt the result of the computation with the sender's private key. The receiver of the message can verify the digital signature with the use of the sender's public key.

In 1978, Rivest, Shamir and Adleman published their paper [RSA78] introducing

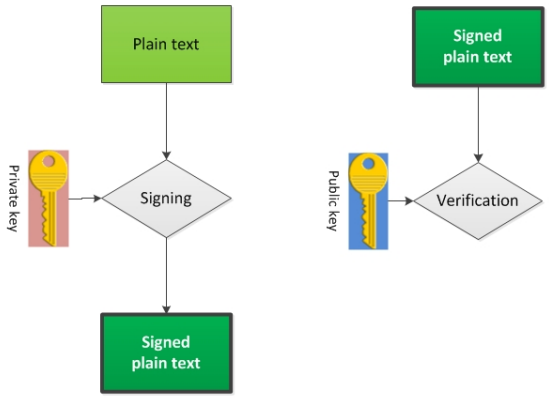

Figure 2.3.: Digital signature with public key cryptography

RSA, the first cryptosystem that could be used as both a key agreement mechanism and as a digital signature.

RSA is based on the hard mathematical problem of factoring composite numbers, called the integer factorization problem. RSA enables the construction of one-way functions with a trapdoor. With RSA Bob can beside creating and publishing a public and private key, also create digital signatures.

For completeness we also want to mention another basis for one-way functions: the difficulty of extracting discrete logarithms. An example of an algorithm based on the discrete logarithm problem is ElGamal. These two problems from number theory are the foundations of most public-key cryptosystem [DK01]. Which algorithm can, is or will be used in the PKI however, is outside the scope of this research and will not be discussed further.

The introduction of every security precaution can introduce other security vulnerabilities, see Sec. A.2. The introduction of the public key cryptography brought along the man-in-the-middle attack. In their famous paper [DH76] on public key cryptography, Diffie and Hellman already pointed out this problem of authenticating that a public key belongs to an entity. The public key cryptography only makes sure the cipher text can only be decrypted using the private key. A third person, Eve, can try to eavesdrop the conversation by giving her own public key to Alice and letting Alice believe that this public key is indeed from Bob. Alice, not suspecting that the public key does not belong to Bob, will encrypt the message for Bob with Eve's public key. Eve will now be able to decrypt and read messages encrypted using her public key. To not unravel her eavesdropping, Eve can encrypt the messages received from Alice again after decryption by Bob's true public key and send them to Bob.

This attack can be, amongst others, prevented by making sure the public key given

to Alice truly belongs to Bob.

Diffie and Hellman introduced in their paper [DH76] a central authority known as the Public File that would serve as a dynamic directory for all of the encryption functions and keys in a system. Each communicant should register their keys there before any communication takes place. In 1978, Kohnfelder [Koh78], proposed, in his bachelor thesis, to have signed certificates by the Public File, instead of saving the information in the Public File itself. This idea of Public File is used as today's Certificate Authority (CA), a trusted third party. Nowadays there are many trusted Certificate Authorities and PKI vendors established.

A digital certificate is a digital signature with the extra property that the message specifies at least a public key of an entity, for which the entity knows a corresponding secret key [Bra00]. The entity signing the certificate, in most cases this is done by a Certificate Authority, guarantees the public key belongs to the entity mentioned in the certificate. If the Certificate Authority is trusted, then the certificate is trusted and the public key mentioned in the certificate can be used to encrypt messages for the entity mentioned in the certificate. Thus, a digital certificate contains the digital signature of the Certificate Authority, issuing the certificate, to allow the verification of the certificate.

The further development of digital communication made the authentication and secure communication important and with the invention of Internet, it became even more important. In the next section we will give an overview of the Public Key Infrastructure and how all cryptography aspects such as public key cryptography, symmetric key cryptography, digital signatures and Certificate Authorities work together.

2.3. Public Key Infrastructure

The Public Key Infrastructure (PKI) is an infrastructure for a distributed environment that centers around the distribution and management of public keys and digital certificates [Bra00] in order to allow secure communication between different parties. In the early 90's the first Public Key Infrastructures were designed and since then PKI has been treated as the answer for users to be sure with whom they are communicating [DK01]. One of the main reasons to develop PKI was the development of Internet and the growing e-commerce, allowing users to purchase products or for online business [Mar11b].

PKI is an essential component on which other applications, system, and network security components are built. The primary function of a PKI is to allow the distribution and use of public keys and certificates with security and integrity [Wei01]. It is important to understand that PKI is not by itself an authentication, application, authorization, auditing, privacy or integrity mechanism but an infrastructure, supporting these and other various business and technical needs.

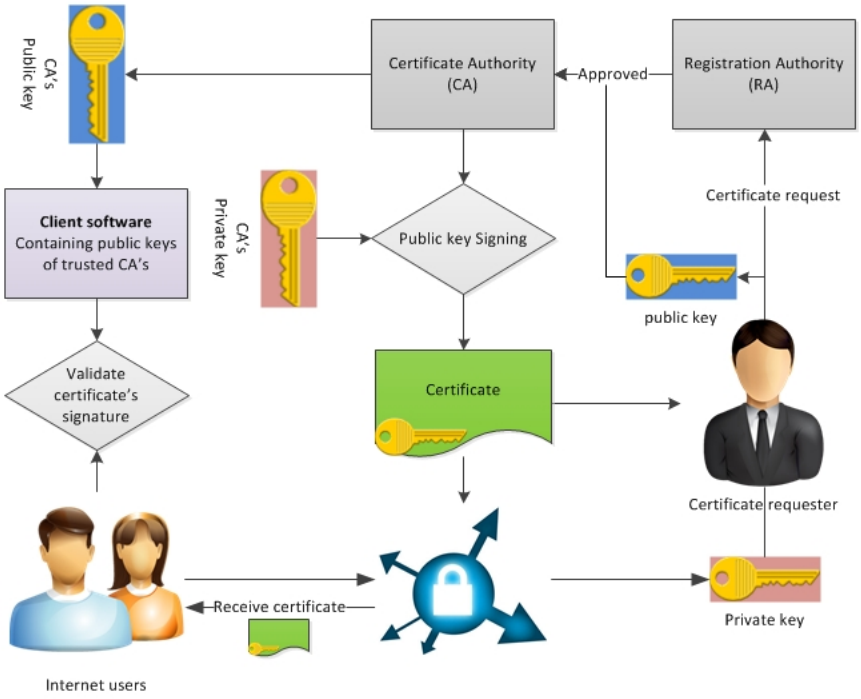

Figure 2.4.: Internet PKI

PKI uses public and symmetric key cryptography in order to exchange a secret key between two parties. The secret key, known to both parties, is then used to encrypt further messages. Simply explained, PKI consists of two important and one optional component: Certificates, Certificate Authorities and the optional Registration Authority.

As can be seen in Fig. 2.4, the usage of certificates starts by the request of an entity. The entity, will create a public and private key pair and request a certificate for some domain at a Certificate Authority. The request will first be verified by the Registration Authority, which is an optional system of a CA, where the CA delegates certain management functions such as registering and checking users that have requested a certificate. After successful verification of the entity's identity, the Certificate Authority will issue a certificate, containing the public key that was provided by the entity, which is signed by the CA's private key. For the interesting mathematical part of certificates we refer the reader to [Bra00, Tel02].

After receiving the certificate from the CA, the certificate owner will place the certificate at the corresponding domain, where Internet visitors will receive it from the domain when connecting to it. Each visitor will verify the certificate before starting a communication with the domain. The verificication can be done when the public key of the trusted CA is known in the Internet user's client software, meaning their browser or Operating System. After successful verification, the user

can start using the public key of the certificate to exchange a common secret key for further encrypted communication.

In Sec. 2.4, certificates will be explained in more details, after which Certificate Authorities and Registration Authorities will be explained.

2.4. Certificates

When Bob wants to communicate securely with Alice, he must be sure that the public key that is sent to him by 'Alice' really belongs to Alice. If Eve succeeds in giving Bob her own public key instead of Alice's public key, from which Eve owns the private key of, she will be able to read the encoded messages from Bob to Alice and sign documents pretending to be Alice. The public-key cryptography is based on the trust that a public key of an identity truly belongs to that identity. Bob can only communicate safely with Alice if he truly is sure he has her public key and not the public key of Eve, the eavesdropper. To give this trust to Bob, Alice can request a certificate from a trusted third party Bob too trusts [Tel02].

A certificate is an electronic document, cryptographically signed by the private key of a trusted third party, containing enough information for users of the open network to verify the identity of an individual, a server, a company, or some other entity and to associate that identity with a public key. The certificate is used to confirm that a public key belongs to a specific individual. In PKI the trusted third party is a Certificate Authority (CA) [Bra00].

Now, when Bob requests a secure communication with Alice, she will send him her certificate first, containing among other information, her public key and the digital signature of the trusted third party. Bob can then verify the public key of Alice by verifying the signature of the CA on the certificate. It is, however, required that Bob knows the public key of this CA and also trusts this CA.

The most widely used format for certificates are those based on the IETF X.509 standards. For the Internet, X.509 version 3 certificate format is used and the infrastructure that provides it is called the public key infrastructure for X.509 (PKIX) [Ben01].

2.4.1. X.509

The X.509 standard defines what information is put into a certificate, and describes the data format for it. There are 3 versions available for X.509. First version, X.509 Version 1, has been available since 1988, and is widely deployed, and is the most generic certificate format. The original problem the X.509 had to solve was access control to an X.500 directory [Gut02].

With X.509 Version 2 the concept of issuer and subject unique identifiers were introduced to handle their reuse over time. This would, for example, allow a new CA to register itself with the same name as an already used, but removed CA name. However, it was recommend by the IETF to not reuse names for different entities, and that Internet certificates should not make use of unique identifiers [CNS+08]. Therefore, X.509 version 2 certificates are not widely used in the Internet.

The current version is X.509 Version 3 from 1996. This version supports extensions which is not needed for a PKI to work properly, but can be important for some organizations to have additional information within a certificate. These additional information is added in a certificate extension and can be information such as policy, usage, revocation and naming data [Wei01]. Some common extensions in use today are: KeyUsage, limiting the use of keys to a particular use only, and Alternative-Names, which allows other identities to also be associated with this public key. In X.509 v3 extensions can be marked critical to indicate that the extension should be checked.

All versions of X.509 certificates consist of the following data.

- **Version:** Is the version of the X.509 used for this certificate, which affects what information can be specified in it. Until now, three versions are defined for X.509.
- **Serial Number:** Is a unique integer in every CA, assigned at the issuing time of the certificate. This is a very important element, because the serial number must be unique for every certificate and is also used when a certificate is being revoked.
- **Signature Algorithm:** Identifies which algorithm is used for the signature.
- **Issuer**: The entity, normally a CA, that verified the information and issued the certificate. Using a certificate from an issuer implies that the issuer is trusted.
- **Validity Period:** All certificates have a period in which they are valid. A certificate validity period can be for a few seconds or for many years. The choice for a period can be dependent on the strength of the private key used to sign the certificate or the amount they client is willing to pay to the issuer.
  - **Valid-From:** The date the certificate is first valid from.
  - **Valid-To:** The expiration date.
- **Subject:** The person, or entity whose public key the certificate identifies. This name should be unique across the Internet and is called the Distinguished Name (DN) of the entity. DN refers to the subject's Common Name, Organizational Unit, Organization, and Country. For example:
  CN = JavaDuke, OU = Division, O = SunMicrosystemsInc, C = US
- **Subject Public Key Info:** Here the algorithm used to create the public key and the public key itself is mentioned.

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

Figure 2.5.: Example of a X.509 certificate for *www.freesoft.org*, generated with OpenSSL

- **Signature Value:** The digital signature computer over the public key of the subject. The signature certifies the entire content of a certificate and not just the public key part.

2.4.2. Certificate Validation

Even though certificates have a validity period during which the certificate is valid, situations can arise where a certificate is no longer trustworthy and thus must be prematurely expired. This action is called the certificate revocation. There are many situations possible that result in such an action. Some scenarios are: the domain may not exist any more, the owner may no longer be a customer, the private key of the owner could be compromised or changed, or the CA could be compromised [Sys04].

Certificates may be revoked by the CA issuing them prior to their expiration time. The action of revoking a certificate must be initiated by the CA or their RA, which also validates the credentials of the certificate requester. Authorities are required to state the way for relying parties to obtain revocation information about certificates

issued by that authority. There are currently two main technologies available to check if a digital certificate is still valid.

Certification Revocation List (CRL)

The Certification Revocation List (CRL) is a commonly used mechanism for relying parties to obtain information about the validity of a certificate. The revocation list is used by browsers, emails and other programs to verify the validity of a certificate. The CRL is a periodically published data structure that contains a list of prematurely expired certificates [Gut02]. Browsers download certificate revocation lists (CRLs) to check the validity of a certificate. This method is copied from the credit card companies who earlier kept a blacklist of no longer valid credit cards [Tel02]. Due to the use of this 'blacklist', the timely publications of the the CRL can be critical for some businesses.

The CRL is timestamped and digitally signed by the issuer of the certificates. Generally a CRL is published within an X.500 directory which also stores the certificates for the particular CA domain.

Since CAs will likely revoke many of their certificates, 10% if they are issued with an intended validity period of one year [Mic02], the CRL will be quite long if the CA has many clients. Browsing performance can suffer when the CRLs get too large. Therefore, the industry has created OCSP, which performs a similar function to a CRL but is far more efficient.

Online Certificate Status Protocol (OCSP)

Another proposal is called the Online Certificate Status Protocol (OCSP). OCSP is currently the newest technology in use to check if a digital certificates has not been revoked. With OCSP, a simple query about the specific certificate is performed, rather than the download of a potentially large list as was the case in CRL. When using OCSP, an OCSP client issues a certificate revocation status request to an OCSP responder for one or more certificates and then awaits for the responder's response to either accept or decline the certificates.

OCSP is a client-server protocol enabling applications to obtain the revocation status of one or more certificates either "good", "revoked", or "unknown". The status is provided by the server either using a real time access to a database or by using CRLs. In the first case, it is possible to obtain timely revocation status information, whereas in the other case, the freshness of the revocation status is not better than the mechanisms it is based on [Pin12].

The CA responds to a certificate status query by generating a fresh signature on the certificate's current status. The key that signs a certificate's status information need not be the same key that signed the certificate [MAM+99]. This reduces transmission costs to a single signature per query, but it increases computation costs. This also decreases security because if the CA is centralized, it becomes highly vulnerable to denial-of-service (DoS) attacks and if it is distributed and each server has its own secret key, then compromising any server compromises the entire system [Mic02].

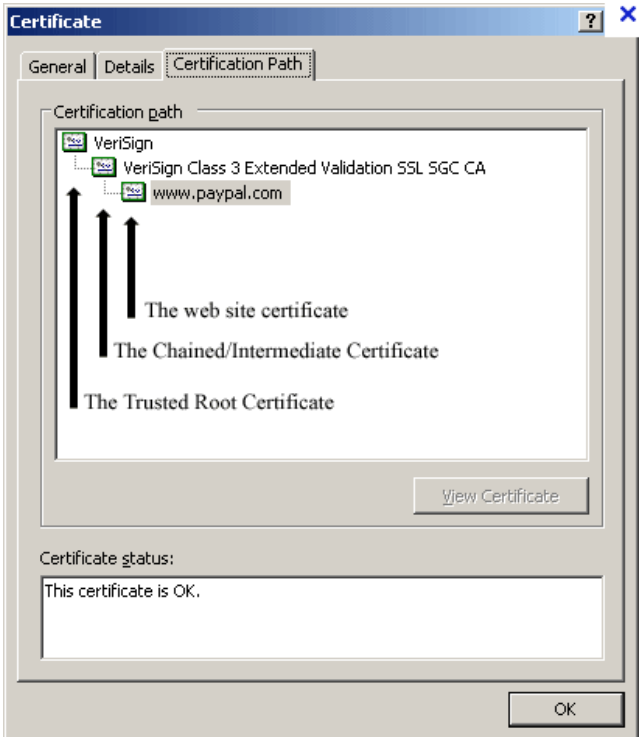

Figure 2.6.: Example of a Certification Path

Privacy however, is an issue in OCSP, since the third party that returns the signed response about the validity of the certificate, knows where the request is coming from and the certificate the user wants to check for validity. Unfortunately, making OCSP privacy preserving is difficult [NST09].

2.4.3. Certification Path

According to the PKI standards, there are two primary types of public key certificates: user certificates and CA certificates.

A user certificate is a certificate issued by a CA to an identity that is not an issuer of other certificates, meaning the identity is not another CA.

A CA certificate is a certificate issued by a CA to another CA. If a Certification Authority is the entity of a certificate issued by another Certification Authority, the certificate is called a *cross certificate* [Ben01]. A list of cross certificates needed to allow a particular user to obtain the public key of another entity, is known as a certification path. A certification path logically forms an unbroken chain of trusted points between two users wishing to authenticate. An example of a certification path for a user, visiting the website www.paypal.com, can be seen in Fig. 2.6.

2.5. Certificate Authority (CA)

In the Internet environment, an insecure and open network, identities do not trust each other sufficient to perform any type of transaction. To provide this assurance, all users or businesses that need to be trusted must have a registered identity. Certificates are used to bind these identities with their private key signed by a trusted third party. In PKI, the digital signature which is used to sign the certificate belongs in most cases to a Certificate Authority. Certificate Authorities are the trusted third parties that generate, distribute and manage certificates and public keys. According to the IETF, a CA is "an authority trusted by one or more users to create and assign public key certificates; optionally, the certification authority may create the user's keys." [Pin08].

In most cases, before a Certificate Authority signs a certificate for an applicant, the identity and authentication of the applicant must be verified. This verification is handled by the Registration Authority (RA). A Registration Authority (RA) is an optional part of a Certificate Authority and acts as an middleman between the certificate requester and the CA, reviewing and approving certificate requests. The role of the registration authority is to confirm the accuracy of the information provided in the certificate request.

There are some policies, dependent on the CA, that the RA must enforce. When a certificate is being requested for a person, the RA validates that the identity of that person is appropriate for the subject name that will be included in the certificate. When a certificate is being requested for a system, the RA validates that the requester is authorized to request a certificate for the system with the specified address (e.g., DNS address) [Ben01].

After a successful verification of the applicant, the CA will generate a signed certificate by the CA's private key, binding the identity to the identities public key.

An insight of the use of CAs is that we are basically moving the problem of trust to another level, see Sec. A.3. Alice must now trust a CA instead of trusting that the public key belongs to Bob. Users do still have the ability to choose, if they want, which CAs they trust. In most browsers, the public key of the trusted root CAs are saved. Root CAs are CAs that can sign their own certificates, whereas the certificate of other CAs are signed by at least one root CA. Users can also add or remove Certificate Authorities from the list. This is also the case in most operating systems [Sto].

Issued certificates by a CA is trusted by end-users as long as the CA and its business policies for issuing and managing certificates are trusted. A CA works within the context of an overall business policy known as Certificate Policies (CP) and functions operationally according to a Certificate Practices Statement (CPS). Other entities, such as higher CAs, will have more trust in a CA dependent on how these CP and CPS are stated and complied [Wei01].

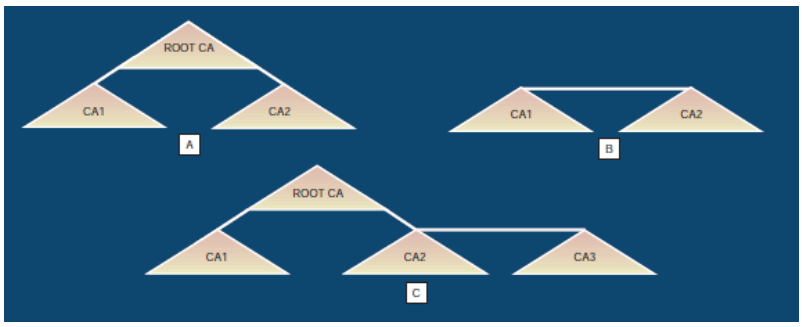

Figure 2.7.: Cross-domain relationships. A: Hierarchical, B: Cross certification, and C: Hybrid scheme

2.5.1. Cross-domain certification

A certificate containing the public key of Alice can only be trusted by Bob if he trusts the CA that signed the certificate. If the CA is not trusted by Bob, he can not trust the public key truly belongs to Alice and therefor will be unable to communicate with her if he wants to communicate over a secure line. However, if one of the CAs trusted by Bob, also trusts this CA, then Bob can trust the public key belongs to Alice and use it for communication. Cross-domain certification is used to create a certificate showing this trust. Previous to a cross-domain certification a CA performs various verifications on the CA it will certify. When a certificate is issued, the trust relationship of a CA is extended.

In PKIs where end entities can trust more than one CA, such as PKIX, cross-domain certification is an important factor.

For PKIX, there are two methods provided for achieving cross-domain certification [Ben01]: Hierarchical and peer-to-peer. Hierarchical implies that most CAs can not create self-signed certificates, but must have certificates issued by a CA higher in the hierarchy, see situation A in Fig. 2.7. In this relationship type, only a root CA can have a self-signed certificate. A user that trusts the root CA, also trusts the CAs that are trusted by the root CA lower in the hierarchy.

The peer-to-peer method, called cross certification, is when a CA from one domain is cross certified with a CA of another domain. As illustrated in Fig. 2.7 in situation B, CA1 and CA2 have issued a cross certificate for each other, showing they trust each other. A cross certificate is an X.509 certificate signed by another Certificate Authority. Note that the horizontal cross certification is not transitive.

2.6. PKI weaknesses

When the first PKI was designed in the early 90's, the Internet was still very small. There were almost no web applications, little to no e-commerce, and probably less

than 10 secure websites on the Internet in 1994 [Mar11b]. There was no information available for designers of how to design such infrastructure and they had no idea the Internet would become so huge. At the time, everyone knew each other and trust was not the main issue. This is not the case anymore, since there were 2,280 millions Internet users in May 2012, where more than 10 million certificates were used on the Internet.

In 2011 two important root Certificate Authority, Comodo and DigiNotar, were compromised. Certificates were fraudulently issued for important websites, both via IP addresses from from Iran [Eck11c].

Here I will discuss the weaknesses of PKI I found through literature research and interviews.

2.6.1. Audit reports

Yearly CA audits are ment to ensure that the level of security at a CA still meets the requirements needed to keep the trust of cross certified CAs, browsers, Operating Systems, and users. If the requirements are not yet met, the audits help the CA to improve themselves. All trusted CAs go through one or more audits, performed by a small amount companies that are authorized for this. The results of the audit is recorded in a report, which the CA can and should be used for improvements. The audit report can also be used to show to other parties to keep their trust in the CA.

However, even though the audit is supposed to result in an objective report, this may not always be the case because the audit company is hired by the CA, see Sec. A.1. The CA is the client of the audit company, one of their sources of income and therefore, simply said, their boss. Therefore, we need to ask ourselves how reliable these audit reports are. In the case of DigiNotar for example, other organizations who also should have audited DigiNotar, trusted that the CA met the requirements, since they had received a positive audit report [Dig12]. Why DigiNotar had received a positive report in the first place is not known. However, this lead to the situation that both OPTA and Logius did not perform any additional audits of their own, which could have improved the situation before the hacker got access to the systems.

DigiNotar has shown that an overestimated value is given to audits. A suggestion to improve this weakness, was to publish the audit reports. However, if the audit report is performed badly or if not everything is reported in the report, then publishing will not help either. Also, if the audit reports were complete, the press could make a big story out of small isuues, see Sec. A.2, distracting everybody from the real problems and giving a bad name to a company that does not deserve it.

Another suggestion would be to have the audits performed by the same company, but that the audit is not engaged by the CA itself, but another independent party, to whom also the report will be given. This may decrease the favouritism in creating true reports.

It must be noted that increasing the audits will not be a solution to the overall problem, since it is only a snapshot of the CA's situation and is not in realtime.

2.6.2. Automatic processes

Almost all processes, regarding the issuance of a certificate, happens automatically, without the interference of humans. An advantage is that processes can be performed much faster and certificates can be requested and issued in less that a day, see Sec. A.1. A disadvantage of automatic operations is that it will eventually become predictable and can be used for an attack against the system, just like the interviewee explained about the MD5 collision attack in Sec. A.4.

It is also not a good idea to have the server or machine that issues the certificates online, as probably was the case with Adobe's servers in 2012 which were used to sign malicious software [Ark12]. Also, Brazilian Trojan bankers got valid certificates in 2012 signed by Comodo for 15 days before they were discovered and revoked [Ass12]! These automatic processes are great danger to the complete PKI security and CAs should pay more attention to whom they give a certificate to. It is a classical case of finding the balance between costs and incomes.

2.6.3. Trust anchor

Trust anchor management is where our trust begins its roots. The trust anchor of the current PKI are the root CAs, trusted by our browsers and operating systems. DNSSEC for example, has a different trust anchor, which is ICANN, the only root browsers and operating systems will need to trust. We can not establish trust without a trust anchor, as there is no other way we can be sure an entity can be trusted. The same trust anchor is seen in our daily life. We can only be sure someone is trustworthy, after we have met the person personally, or if we know the person through someone we already know.

The DigiNotar incident exposed an important issue regarding trust anchor management. It is argued that the current methods of embedding trusted certificate lists in applications, browsers, and operating systems are far from perfect.

Most users are currently not vulnerable to the fraudulent certificates from DigiNotar, because all known fraudulently-issued certificates have been revoked and the most widely used browsers and email clients have been updated to distrust DigiNotar's root CA. An issue here is that users employing outdated browsers and operating systems may be still at risk, particularly if their Internet Service Providers will conspire to divert their web requests to a spoofed server [Gam11]. The proportion of users that dont yet have installed the most recent version of their browser is 19.8% of Internet Explorer, 20.8% for Chrome, and 33.9% for Firefox [Ley12].

Unfortunately, there is little we can do about users using outdated software. This practice of updating has been used for a long time, and until now, there has been little evidence of more effective implementations. There will always be some security issue that will need a patch. Users have to update their software to be protected against at least the known and patched security issues.

2.6.4. Trust in CAs

Another issue, very similar to the trust anchor issue, is that we can not be sure about the liability of a CA that we now trust, in the near future. As mentioned before, a trusted CA by browsers and operating systems, can simply issue any certificate for any domain in the world. A country or a person for that matter, could also buy or create a CA, or an intermediate CA, that will be trusted by most browsers for only a couple of thousand dollar.

> *"Nothing is more destructive of respect for the government and the law of the land than passing laws which cannot be enforced."* Albert Einstein

The China Internet Network Information Center (CNNIC) is for example a trusted root CA, see Sec. A.1. The CNNIC is said to be controlled by the Chinese government and is alleged to be heavily involved in spying on Chinese citizens. Even if the CA organization acts as a fair CA in the beginning, this does not mean that the organization will stay honest. As the previously used real life example, a person we know now, can prove to be very different in time. Logical conclusion: Trust is for sale!

2.6.5. The weakest link

One of the most important design weakness of the current PKI is that any trusted CA, can issue certificates for any website, application, email etc. For Mozilla, there are more than 120 trusted root CAs. In Windows 7, there are only 19 root CA included in the trusted list [EB10]. The 19 CAs may sound little, but the reader should remember that each root CA is able to grant another CA, called intermediate CA, the ability to also issue certificates, or even grant other CAs to do the same.

In the end, the amount of truly trusted CAs grows notably, where, compared to other CAs, there will always be some trusted CAs with the weakest security. Naturally, one of the weakest links will eventually be successfully attacked. Only one CA with a certification path to a trusted root CA is needed to be compromised in order to allow the attacker to issue fraudulent certificates.

This can for example happen when Eve issues a certificate signed by the private key of a compromised trusted CA. The issued certificate then no longer contains the correct public key of Alice, but contains the public key of Eve, the attacker. Now, Eve can use the fraudulently issued certificate by performing a MIMT attack to

divert Bob's communications and send Bob the fraudulent certificate, instead of the certificate containing Alice's true public key. who wants to communicate with Alice. Since Eve's fraudulent certificate is issued by a trusted CA, Bob's client software will verify the certificate as valid and show no warning messages. After verification, Bob thinks he can securely communicate with Alice, where in fact, he will agree upon a secret key with Eve. Eve, in possession of the private key pair, will then be able to decrypt the secret key and continue communication with Bob.

2.6.6. Competition between CAs

CA companies charge clients for issuing a certificate. Since there are many CAs available nowadays, they compete with one another in the price. One simple way for a CA to save time and reduce costs is by simplifying or removing the verification process of the requesters identity. This already has resulted in the negative outcome that anybody willing to pay for a certificate, can get one. At the RSA conference in January 2000 Matt Blaze said that: "A commercial CA protects you from anyone whose money it refuses to take" [ABD+00].

Lately in the news [Koe12], there was a discussion about whether CAs also should check the reliability of a requester. An example of this is that a certificate should not be issued to the owner of the domain www.abmamro.nl. However, CAs replied that this was an impossible task to perform. What they did not mention, was that this costs too much to check, while issuing such certificate is an income.

3. Companies and certificates

In the previous chapter, PKI and its weaknesses have been explained. In this chapter, we take another point of view by examining the problems organizations can face by using PKI and how these problems could be minimized or eliminated. This section has come into existence after noticing, through interviews and literature, that companies are not enough aware of the risks of using certificates and the measurements they can take to protect their company against compromisation of a CA.

Organizations are often not aware of the importance of their certificates. They take no measurements to prevent the possible problems that can arise in case an incident happens, until it is too late. In Sec. A.3, the interviewee said the DigiNotar incident made their company more aware of the situation, because they had to make an overview of all the certificates they used in order to be able to conclude whether they used DigiNotar's certificates. Making such overview is not an easy task for a company that uses many certificates. Although the company of our interviewee is now more aware of the importance of a CA incident, this has not led to actions preventing the impact of future incidents. Fortunately, there were no certificates issued by DigiNotar for their company or collaborating third parties. If this was the case, the company had to find a replacement for the compromised CA as soon as possible, and reissue those certificates by the new CA. This would not have been an easy task after such incident, and specially not if other companies get a priority to replace their certificates due to for example, previous agreements.

A security incident at a CA can have great consequences for the CA itself, its users, or companies using certificates signed by that CA. If false certificates are found and discovered, the trust in every certificate can be questioned and various parties will start revoking their trust in those certificates. In the worst case, the CA responsible for the false certificates will be removed from their list of trusted CAs, resulting in the distrust of all certificates signed by the CA, which will end all businesses for that CA. This happened for the first and only time in 2011 to DigiNotar [Dig12].

A certificate incident can have great consequences for a company if no measurements are taken. A company, in possession of a certificate signed by a CA where the incident took place, will need to replace all of these certificates. In this chapter we will describe the risks, the important points companies should pay attention to when certificates are used and the actions they could take to minimize or prevent incidents.

3.1. Risks of certificates

Companies use certificates to guarantee their authenticity to clients or other companies. It is mostly used for either websites, documents, programs, emails, or other electronic messaging systems. The certificates are meant to give more trust about the source, but also integrity about the information to the one receiving them. As explained before, as soon as a certificate incident is discovered, the compromised certificates will be revoked or in worse situations, the CA's certificate, signed by a root CA, will be revoked. If the CA itself is the root CA, it can be removed from the trusted root CAs list, which happened to DigiNotar. Compromised certificates can be the consequence when a CA, RA or when the company itself gets compromised:

Compromised CA A CA or its intermediate CAs, can get compromised. This may happen when attackers get access to one of the essential systems, containing for examle the Certificate Revocation Lists (CRL) or the program issuing the certificates. Attackers may also obtain the private key of the CA, which could be used to sign their own fraudulent certificates or worse, their own intermediate CA. In 2011, one of the resellers of Comodo was compromised, resulting in the issuance of fraudulent certificates which were signed by Comodo and were therefore valid certificates, see Sec. A.1 and [Com11a]. Besides attacks from the outside, a CA or its resellers can also misuse their power to issue a fraudulent certificate for a domain. This may also happen if the company is under influence of the government.

Compromised RA When an attacker would get access to the RA, the attacker could allow the issuance of fraudulent certificate requests, resulting in the issuance of that certificate by the corresponding CA. An fraudulent certificate can also be issued when a malicious user convinces the RA to issue a certificate in the name of another or non-existing company.

Company's private key The private key of the company can get stolen, which if happens, the messages sent to the company by clients, using their valid certificate and 'trusted' public key, can be intercepted and decrypted. Attackers can decrypt the messages since they posses the private key. This can occur by a physical break-in into the company, a brute force attack on the company's systems or by malicious software that can get inside important systems though multiple ways. When the private key of a company is compromised, the company needs to generate a new private and public key pair and get their CA to issue a new certificate for the new public key, and revoke the previous certificate. If the previous key pair was obtained by a software attack, the company also needs to find a patch for this.

After the discovery of one of these incidents, the compromised certificates can be revoked by the corresponding CA through a CRL, the online certificate status protocol (OCSP), or the Trusted List, in case it is for software, which happened to Adobe in October 2012 [Ark12]. Revoking certificates also involves technical actions such

as updating browsers, operating systems, and other software. A revoked certificate can lead to security warning messages, disturbed communications and electronic documents from which the authenticity and integrity can not be guaranteed anymore [Cen12]. In Sec. A.3, the interviewee said that after DigiNotar was not trusted anymore, their company had to do a lot of research in order to get an overview of their certificates to find out whether they actually used a certificate issued by DigiNotar or not. Other companies were less fortunate and needed to to replace their certificates as soon as possible.

3.2. Measurements

In this section we will discuss measurements to prevent certificate incidents. Organizations need to establish formalized plans to minimize the impacts of a certificate incident, which were discussed in the previous section. Four different categories of measurements will be discussed here:

Prevent Measurements to minimize the risk of incidents

Detect Measurements to detect incidents as soon as possible

Limit Measurements to limit the consequences and damages after an incident.

Correct Measurements to repair the damages after an incident has happened.

Even though some measurements are to minimize the risks during or after the incident has happened, the measurements still should be prepared before the incident. A well prepared organization will take these measurements in advanced in order to minimize the risks and allow fast recovery.

3.2.1. Prevent

Overview

Many organizations do not have a good understanding of their inventory of digital certificates or where they are. It is necessary for a company to have a good overview of all of the in use certificates inside the company. While several offerings exist to discover X.509 certificates, most organizations rely on spreadsheet-based tracking methods and manual processes to keep track of certificates, resulting in many undocumented installations and increased exposure to risks [OW11]. In the overview of certificates, the CA, RA, CA path, the use and application of the certificate, the name and contact information of the administrator for each certificate should be mentioned [Cen12]. It is also clever to mention if there are third party certificates necessary for the company. Several commercial certificate authorities offer Certificate Management System (CMS) software, which is a networked system for discovering, identifying, tracking, notifying, and ultimately automatically renewing and auditing the installation of new certificates [OW11].

Standards

Companies should be aware and know the standards that are used and not to get surprised by updates of others parties, where the company depends on. In Sec. A.3, the interviewee said they had a little bit trouble when Microsoft announced that keys under 1024 bit would not be accepted anymore. To prevent this, companies could continue to stay up-to-date by keeping up with the current standards. An example of such standard are papers published by The National Institute of Standards and Technology (NIST), see [BBB+07].

Private keys

One of the reasons why certificates get compromised, is because the private key of the company gets stolen or is exposed to malicious parties. A company should make sure their private keys are stored carefully. A good idea is to keep the private keys offline in hardware security models (HSM), which provide strong authentication to access the content. If a back-up of the keys exists, they should also be securely saved.

3.2.2. Detect

The later a company takes actions, the more they are exposed to possible problems and brand damage. Companies can scan their own networks, systems and applications for possible intruders, check the website of CAs for possible incidents and immediately start actions to limit the problems. It is worth noting that if another CA is compromised, attackers could issue a certificate for themselves in the name of any company and any website. There is little a company could do about this except reporting such certificate if found. This problem should be fixed in PKI itself.

3.2.3. Limit

Companies could spread the risks by requesting certificates from different CAs, under different root CAs. This can limit the risks in case an incident my happen and one of the CAs gets compromised or gets distrusted. This also gives the company the possibility to be able to replace the certificates from the compromised CA faster, because the company is already registered and approved at another CA.

3.2.4. Correct

If finance is not a problem, companies could make agreements with another CA to allow faster issuance of certificates for the company in the case an incident would occur. Another possibility is to request a backup certificate from another CA that allows the company to immediately start replacing certificates when an incident has

occurred. Note that it is very important to have a backup CA or certificates from another root CA than the CA that is currently used by the company. This prevents the chance that the compromisation of the root CA affects both the current and the backup certificates.

4. Game companies

In this chapter, the security of online games is discussed. Through interviews and literature study it turned out that cheating is the most important security issue in online games, which has become the focus of this sub research. The goal of this sub study is to find solutions against cheating in online games, possibly by PKI. We begin this section with a brief classification of cheating in Sec. 4.1, which is used in Sec. 4.2 to give provide possible solutions per classification. For more attacks and cheats the reader is referred to [NPVS07, YR05, WS07].

Traditionally, each player played a game on his own device, PC or a console, possibly with artificial intelligent characters. Different from the traditional games, online games allow much more users, who may be in different places over the world, to play together over the network, allowing many players to interact with each other in the same world. This is resulting that our world is going towards the era of online games, where millions of players spend much time and effort into a game.

Research has suggested that cheating is a major security concern for online computer games [YC02]. A minority of cheaters can potentially ruin the game for all players and must be prevented for an online game in order to be successful [WS07]. Cheaters can for example can get the same rank or score for less time and effort.

Cheating is defined in [ALRL04] as *"Any behavior that a player uses to gain an advantage over his peer players or achieve a target in an online game is cheating if, according to the game rules or at the discretion of the game, the advantage or the target is one that he is not supposed to have achieved".*

Not only successful games like World of Warcraft, which attract millions of players and create considerably large revenues, have problems with cheaters. In the gaming industry, there are also many relatively small game studio's compared to the big game studios such as Blizzard, Valve or EA. The smaller game companies also experience difficulties in their games caused by hackers. However, the only priority these small game studios have, is to have their game played, because that delivers revenue. Most of the security implementations of these game studios, depend a lot on the security measurements of their publishers or platforms, see Sec. A.6 and Sec. A.8. These small game studios do not have the time, personnel or enough knowledge to implement their own security measurements. They use the included libraries by their platform or use standard solutions to make their game more secure. For some cases these solution work well enough. For example, games published on the platforms seem to have less troubles caused by hackers, see Sec. A.8 and Sec. A.7. This is

however not due to the great security of the standard platform's library, but has more to do with the difficulty to attack the game in the first place. This difficulty directs the hackers towards PC games. Hacking a game on the PlayStation for example requires the hacker to first decrypt and modify the CD. After the modification of the CD, extra steps are needed to make sure the PlayStation can read it again. After many steps, the hacker can actually start looking into the game, see Sec. A.7 for the reasoning of a game hacker. Nevertheless, it is a good idea for any game studio to take the vulnerabilities and possible solutions into account and weight that against the costs in order to make a decision.

For the scope of next sections, where the results of interviews and literature study about the security vulnerabilities and solutions to secure online games are discussed, we will first define some terms. The user of a game is called a player. The player interacts with the game through a game client. The game client communicates with a game server or other game clients. A game server controls the game and interacts between other players.

4.1. Cheating classification

Yan and Randell [YR05] provide an extensive list of cheating techniques, and formed a taxonomy with regard to the underlying vulnerabilities, consequence and the cheating principle. They classify the vulnerabilities into two divisions: system design inadequacy, which concerns a technical design flaw that arises during system development, and people vulnerabilities, involving those who operate or play online games. Because a cheater can exploit a flaw in a game system, a flaw in its underlying networking or operating system, or both, system design inadequacy is divided into two subdivisions: inadequacy in the game system and inadequacy in the underlying systems. The authors also introduce fairness as a vital additional aspect to the traditional security aspects confidentiality, integrity, availability and authenticity and argue that fairness and its enforcement appear to be a proper perspective for understanding the role of security and cheaters in online games. Although Yan and Randell are large and detailed in their taxonomy, their characterization of cheating lacks structure, and it is argued that new forms of cheating cannot be easily integrated [NPVS07].

Neumann et al. [NPVS07] distinguish three categories of cheating based on the threatened game property: confidentiality, integrity, and availability. Confidentiality requires that a cheater dos not have access to state information they are not allowed to. Integrity ensures that a cheater cannot modify the game state or its fundamental laws. Availability ensures that the game is available to all players at all times. Unfortunately, this paper only provides a brief discussion of possible cheats and their methods of attack.

GautheierDickey et al. [GZLM04] proposed a cheat classification scheme comprising four categories: game, application, protocol, and network. Game cheats do not

require any external programs or modification and occur entirely within the game; application cheats require using or modifying applications; protocol cheats interfere with the game's communication protocol; and network cheats involve modifying the network infrastructure over which the game traffic is sent. The authors do not consider all forms of cheating [WS07] and therefore their classification is not complete.

Webb, et al. [WS07] modified the classifications of GautheierDickey et al. into: game, application, protocol, and infrastructure. Infrastructure cheats involve modifying or manipulating pieces of infrastructure that the game relies on, such as drivers, libraries, hardware, the network, etc.

We continue with the classification of Webb by dividing the vulnerabilities and solutions of online games into game, application, protocol, and infrastructure.

4.2. Possible solutions

As the game hacker said in Sec. A.7 and was explained in [MGM06], as long as the hacker is in possession of the game software, the hacker has enough time to figure out how to abuse this information. Any protection mechanism will therefore eventually not be reliable if it is integrated into the game client.

Although it seems like a lost battle for the game developers, the key point is to keep the balance between time and effort the hacker needs to hack the game and the time and effort the game developers needs to secure the game, see Sec. A.8. The amount of revenue or players certainly play a big role in both sides. The time and effort one would want to spend on the game, on both sides, increases as the amount of players grow.

The game hacker I talked to said he had not even tried to find out the weaknesses of games run on PlayStation or Xbox, because getting through the platform in the first place, is a big barrier which costs too much time and effort, see Sec. A.7. The game studio with the online multiplayer game also said that most of their security issues originated from the PC and not the Playstation or Xbox, see Sec. A.8. Although this can be the case because most of their players were from the PC as well. Since a minority of cheaters can ruin a game, it is necessary to pull the balance towards the game developer's side, as far as it is allowed by the game revenue.

4.2.1. Game level solutions

Game level vulnerabilities occur completely within the game program without any modification or external influence. Cheating in the game level does not require an understanding of the game files or any need for programming skills.

Bugs in the design or implementation that can give players advantages belong to game level cheats. An example of bugs occurred in Halflife, where a specific combination of actions allowed cheaters to re-load weapons faster than honest players [Pri00].

It is believed that this failures and bugs originate from the commercial pressure of game development, which ensures that there will never be sufficient resources available to properly engineer security, and that cheat developers will always have the advantage [DKL+04]. Therefore, although it is very hard to prevent all game bugs, it is necessary that bugs can be reported by players and detected and patched as soon as possible. In some cases, a database rollback may be required if the influences of the cheat significantly impacted the game.

There also exist some anti-cheating products, for example Punkbuster and Cheating Death, which work by installing them on the game client side. The anti-cheat verifies the client game code and detects certain patterns of cheating, making cheating harder. However, as stated before, any protection mechanism on the client side can eventually be abused by hackers.

Another game level cheat is when for example two or more players play on the same account to get faster results or when a low paid worker, usually in China, is paid to play the game and earn valuable items, which then is sold to players [Lee05]. The latter often occurs in World of Warcraft and players are regularly banned for such a behavior. While the used method by Blizzard to detect these cheaters is unknown, it is suspected that they use statistical analysis of log files generated by the servers [WS07]. Laurens et al. showed in their paper [LPBC07] a cheat detection design that statistically analyses server-side observable behaviour for indications of cheating.

A simple, but rather difficult attack to prevent is collusion, where two or more cheaters work together to gain an unfair advantage. Colluders often communicate via an external channel, such as Skype or TeamSpeak. In [Yan03] several approaches are proposed to prevent collusion, such as monitoring the players using a webcam

4.2.2. Application level solutions

Application level vulnerabilities, unlike game level vulnerabilities, require either the modification of the game executable or data files, or running programs that read from or write to the game's memory while the game is running. Developing application level cheats requires knowledge of reverse engineering.

An example of application cheats is the creation of bots. One such bot is the fish-bot in World of Warcraft that catches fish, delivering the player some gold for each fish the bot catches [Fis]. One way to prevent this is by using anti-cheat software such as PunkBuster or the Valve Anti-Cheat, which is also used for steam games, see Sec. A.8. These programs can check the running processes in the memory against a

database of known cheats to detect bots. Another way is to use statistical analysis [YLLY06], which is probably used by WoW, see Sec. A.7. However, this can possibly be bypassed by introducing randomness into a bot [Pri00].

Another important attack is the modification of the game client, which could deliver the hacker benefits in the game. A simple solution would be to obfuscate the excecutable in order to make it harder to read for humans. Obfuscation, however, is not a bullet proof solution. As explained in Sec. A.8, obfuscation has consequences for the performance and in some cases the executable will even be seen as a virus by some anti-virus programs. Simulating and verifying all commands of the players by a server or referee that can be trusted to produce correct results, as used in the game League of Legends by Riot Games, see Sec. A.7, is another solution to dissmiss cheating by the modifications to the game files.

Another solution, which nowadays could work because of the fast Internet connection of most players, is to have the complete game or at least the essential parts online and send the information only to the player when needed. The player would then only need to send the key strokes to the server or other clients. Such mechanism is suggested by Mönch in [MGM06]. The approach prevents an attacker from modifying a game client and from accessing sensitive information in the game client's memory.

While this solution eliminates a number of cheating possibilities, it puts a high load on game-servers. In online games, scalability of game-servers is already a major problem [CXTL02]. Most games will therefore not be able to use this method and need the player's computer to take some computation weight off their shoulder. It is however possible to perform the most important calculations alongside the player's computer to check the player's calculations. This, however, will need a trusted server and will solve the problem that players cheat in the global score.

Another solution, similar to the previous, but does not require a trusted server, is to replicate the complete game in all of the clients. These clients form a peer-to-peer network and all clients process the same events. A Peer-to-peer based game topology promises to reduce the cost overheads and reliability issues of running complex and expensive central servers. The complexity of the game, however, is limited to the player with the least computer calculation capacity. Another problem that arises with checking all the other clients in a peer-to-peer network is that a hacker now also can obtain infomation of other users he is not allowed to have [Pri00]. Also, this does not disallow the hacker to setup an own server where players can play the game. This can be done with games, such as World of Warcraft or Eve Online, that need a monthly payment to be able to play the game. As long as the hacker is able to get the code of the game, this can not be prevented. A non-security solution would be to make it more fun to play on the original server than a private server by creating special events in the game Sec. A.7.

4.2.3. Protocol level solutions

One of the Protocol level vulnerabilities is the interference with the sent and received packets for the game's communication. Packets can be inserted, destroyed, duplicated, or modified by an attacker. This can be done using for example Wireshark, see Sec. A.7 or any other network interception program. A possible attack for the protocol level could for example be the modification of the level of health or strenght of the player. Other attacks include the delay of packets in order to receive information from the opponents before sending the hacker's own actions.

A solution to delayed packets is to have dead reckoning calculating the next couple of positions of the player according to the current values with the addition of denying delayed packets. This solution, however, has the disadvantage for players with slow or lossy Internet connection [WS07].

Encrypting the data using public key cryptography or symmetric key cryptography is always a good idea to secure packets. Sending raw data makes it simply too easy for a hacker to modify. When packets are encrypted, the effort of the hacker increases. He will first need to reverse engineer the code and find out how the packets are encrypted, then try to decrypt the packets. However, this can have consequences for the performance of the game, see Sec. A.5, and eventually the hacker will be able to determine how to decrypt the packets and modify them.

Symmetric encryption is a better choice and could be realized when both parties have agreed on a secret key. This could be done using the public key cryptography or by using an authenticator, which is used by Blizzard for their popular game: World of Warcraft. An authenticator, which is a variant of what is also used by banks, is an additional security measurement to log into the game. A Cheater wanting to acces the account of another player must then obtain not only the password, but also the unique code that is generated by the authenticator. The authenticator generates a unique code for every login. For more detailes about the working of the authenticator, see Sec. 4.3.1.

Figure 4.1.: Blizzard Authenticator

4.2.4. Infrastructure level solutions

Infrastructure level vulnerabilities is the modification or interference with the software or hardware the game is using. An example for infrastructure level cheats is the modification of the graphical drivers to render the world differently, such as with transparent walls which allows the cheater to see behind walls, receiving information the cheater is not supposed to have, such as the opponent's position.

By using an anti-cheat program, such as Punkbusters, screenshots can be made of the players screen, which is one way to detect wallhacks. Another solution would be On Demand Loading. Using this technique a trusted server can store all secret information and only transmit information to clients when needed. Therefore, the client does not have any secret information that may be exposed [LDMW04].

4.3. Games & PKI

Research shows that Chinese gamers had spent 1.7 billion dollars on games in 2007, a staggering amount that is expected to reach a figure of 6 billion by 2012. Traditional online games still have a substantial customer base, but MMOG are becoming ever more popular. They contribute a significant percentage of revenue to the gaming industry [Vas12].

In our interview with a game developer at a game studio which has created an successfull online multiplayer game, see Sec. A.8, it was explained that they used public key cryptography to encrypt the player's communication, but not to identify players. This was not even possible, since the public and private keys were generated every time the game was executed.

To establish idetification, their used method could be expanded towards PKI, where users would be sure they are communicating with a valid server or validated game clients. However, as explained in previous sections, we can now fully assume that the client side will eventually be changed by hackers, who would be able to find out exactly what the private key is. Also, having fixed public and private keys will probably make it even easier for the hacker, because now the keys are static and are not regenerated randomly every time the game starts. Finding out the key once would then be sufficient.

If we really want to use public key cryptography effectively, we must hide the key someplace where hackers have no access to. This could be done by storing the key outside the game client, in a keytoken kind of hardware. Of course, even keys hidden in hardware can be hacked, but the procedure for it is much more difficult, time consuming and has higher costs due to materials. An real life example of this method is the Authenticator for Blizzard's Massive Multi Player Online Game: The World of Warcraft, see Sec. 4.3.1.

The Authenticator is used to make sure that user accounts are only accessed by the user itself. The Authenticator, similar to the one Vasco creates, can be seen in Fig. 4.1. The idea is that the authenticator generates an output that is combined with a time stamp, which can only be used for one authentication per time. The authenticator uses symmetric key cryptography, which will be a problem if Blizzards servers, containing these keys, would be compromised. This will be a huge loss for all players and Blizzard, because every key token then would not be trusted anymore and should be replaced.

Similar attack occured already in August 2012, where access to Blizzard's internal network was gained and some user data was stolen [Mor12a]. Using public key cryptography instead of the used symmetric key cryptography would prevent this problem, since attackers would only obtain public keys of the Authenticators, giving them no extra information.

4.3.1. Authenticator: How does it work?

The idea of the authenticator is to setFigure up a second channel through which Blizzard can prove the identity of the player, this is called a "two-factor authentication", and combines something the player has, the authenticator, with something the player knows, like a password, on the assumption that it is difficult for an attacker to take both at once.

Similar security measures are used to protect banking or govermental systems around the world. If someone cracked the algorithm behind the tokens, they would probably be using this information for something more important than WoW accounts.

Each authenticator contains three things: a clock, a secret key, and a function which takes the time and the secret key and generates a number. The secret key is not necessarily the serial number. The function is well-known; the secret key is not. Each key is valid for about 30 seconds, after which point the authenticator function returns another key. Blizzard accepts one or two codes in either directions, in case the player's authenticator has a faster or slower clock or the player types too slow. Important is that each key only works once: once used, Blizzard does not allow the player to log in with that key again. In short, stealing the authenticator code only gives the hacker one login within the next minute or so.

To be able to validate the generated keys by the player's authenticator, Blizzard or their partner Vasco, has a big database with the serial numbers of every authenticator token ever made and the associated secret key. After buying an authenticator, the player must link the authenticator to the desired account. After the account is linked to the authenticator, the player can only log in using a password and the key the authenticator generates after pressing its button [Cil12].

The price of an authenticator is around 10 Euros in Europe, with a battery life of approximately 7 years.

5. Possible PKI solutions

5.1. Overview

PKI is designed to assure that the public key of an entity truly belongs to that entity. We do this currently by putting this trust in the hands of third parties, called Certificate Authorities (CAs). Most of the trusted CAs are very secure, but it can happen that one of the many CAs gets compromised. In the past year, four Certificate Authorities have already been compromised [Eck11c], with DigiNotar and Comodo being the most important two of them. As Eckersley wrote: *"Each of these incidents could have broken the security of any HTTPS website"* [Eck11c].

The problem we try to address here is not to the problem of a CA getting compromised, but that when this happens, the CA, if trusted, can issue fraudulent certificates for any domain. Next problem is that the discovery of the compromisation can take some time, because the issued fraudulent certificates, can only be checked by the CA itself and observant users who are already being attacked using the fraudulent certificate. The latter is how the first mention of DigiNotar's fraudulent certificates was made [Pri11]. The only authority that can perform this check beside the CA itself, are companies performing the audit at the CA. These companies are hired by the CA and do not provide a continuous solution, extending the time of discovery, see Sec. 2.6.1. Also, the CA is aware of the audits and could misuse its powers just after the audit has been performed.

As there are many solutions that try to solve some problems of the current PKI, described in Sec. 2.6, a selection of these solutions had to made, since discussing them all would exceed the size of this research.

Therefore, only solutions that have attracted the most attention during the past year by our interviewees or Electronic Frontier Foundation (EFF), have been selected. The EFF is an international non-profit digital rights organization since 1990, standing for free speech, innovation, privacy, and transparency. Beside their many actions, they also support new technologies which they believe preserve personal freedoms.

Public Key Pinning, Perspectives & Convergence, and DANE have been mentioned by the EFF [PE11] as a possible solution, where Sovereign Keys is proposed by the EFF itself. Certificate Transparency is an idea, inspired by the Sovereign Keys and created by the same authors of Public Key Pinning.

Public Key Pinning and Perspectives & Convergence even have established working code and are currently being used. Next to these 5 proposals, a new proposal, named

the Multiple Certificate Signatures (MCS), is added by me. MCS is meant to make the current PKI stronger and robuster by signing a certificate by multiple Certificate Authorities instead of one. The idea is to remove the single point of failure, which with the current PKI, lies at the Certificate Authorities.

Other solutions, such as PGP, CAA records, and Whitelisting, are not discussed extensively, because either one of the discussed solutions is a better candidate or the solution has not the capacity to remove or limit the risks of an compromised CA.

Pretty Good Privacy (PGP) uses a distributed Web of trust where no trusted third parties are involved that actually couch for the identity or integrity of the certificates. This kind of trust model does not scale well for the Internet, since each Internet user would have to determine its own level of trust it will accept from other entities [Wei01]. This would hinter the fast communication that is now expected for e-commerce. For PGP holds that Perspectives & Convergence are based on the same idea, where Perspectives & Convergence seem to be its improved version and will be analyzed in Sec. 5.3.

CAA record's idea is similar to DANE, but does little to prevent or detect already fraudulently issued certificates, where instead, the objective is to reduce the risk of issuing fraudulent certificates [PHBL12]. Since this is not the improvement this research seeks, this solution is not discussed further.

Whitelisting, an easy solution that might solve the problems, is simply the opposite of the current black listing method, where we can conclude that if a certificate is on the black list, it will not be on the whitelist. As the interviewee in Sec. A.3 mentioned, whitelisting will not work when managed by the same CA, where processes are automated and therefore, when compromised, modifying the whitelist will be as easy as modifying the blacklist. Furthermore, adding an extra layer of security, which whitelisting would be if it would be completely disjoint from automatic processes, does not yet solve the problem that if a CA gets compromised, they can issue certificates for any domain.

For every solution, first the idea and its working, is described, followed by an analysis of the yet to be solved problems. An overview is then given using the criteria described in Sec. 1.3.1.

Even though we have set criteria in order to measure the solutions against each other, it is not my intention to point out a solution here as the best solution. My intention is only to point out the weaknesses that yet need to be worked out. It is important to understand that even though a solution may score bad now, it may be the best solution when the weaknesses are solved. This chapter should be treated as a challenge in order to solve the issues in the existing solutions and should not be seen as bashing the discussed solutions, but merely as criteria to help them improve.

The criteria and analysis are performed from the point of view that the solution would replace the current CA system. Using this assumption, it is easier to un-

derstand and decide which of the solutions will have a chance of blooming into a solution that could replace or improve the current Internet PKI.

5.2. Public Key Pinning

Public Key Pinning allows domain holders to 'pin' their domain to a CA. The pin ensures that only certificates from the pinned CA will be accepted for that domain. This would limit the risks of accepting fraudulent certificates from other compromised CAs for that domain.

The idea is that browsers will take care of the pins and therefore know which CA is authorized to issue certificates for a specific domain. Websites can tell browsers via an HTTP Strict Transport Security (HSTS) header which CAs should be trusted to issue a certificate for their domain names. These 'pins' are then remembered by browsers. Whenever a certificate is then showed to clients, the issuing CA of that certificate must correspond to the CA in the pin. A valid certificate, issued by a trusted CA, but other than the CA in the pin for that domain, will generate an error and deny the secure connection request.

The solution was introduced by security engineers from Google, who also wrote the proposed Internet-Draft called "Public Key Pinning Extension for HTTP", which was published in November 2011. In this draft, they proposed an extension to the HTTP protocol, called HTTP Strict Transport Security (HSTS) [HPJ12], allowing domain owners to "pin" the domain's cryptographic identities for a given period of time, which is indicated by the max-age of the pin. The domain's cryptographic identity is the fingerprint of the SubjectPublicKeyInfo (SPKI) in the validated certificate chain. The SubjectPublicKeyInfo is the public key in the certificate and the algorithm with which the key is created [CNS+08]. During the period the pin is alive, the domain must present at least one certificate chain whose fingerprint matches at least one of the pinned fingerprints for that domain [EP11].

The reason for remembering the public key (SPKI's fingerprint) instead of the certificate is because there can be multiple certificates available at the same time, for the same domain and public key, with different extensions or expiry dates. For example, StartSSL has two root certificates: one signed with SHA1 and the other with SHA256 [Lan11e].

A more static implementation is already being used in Chrome for small amount of domains, including Google's own domains. In May 2011, Adam Langley said in his Imperial Violet blog: *"The whitelisted public keys for Google currently include Verisign, Google Internet Authority, Equifax and GeoTrust. Thus Chrome will not accept certificates for Google properties from other CAs." [Lan11e].*

The advantages of this solution is that the risks of compromised CAs will be limited to only domains with pins for the compromised CAs. Thus, compromisation or abuse

at one of the CAs, that is mentioned in the pin, can still result in eavesdropping the users. However, this risk is significantly much less if a compromisation occurs at one of the 600 trusted CAs in the world and limiting the risk is indeed our goal.

5.2.1. Analysis

Public Key Pinning is also what allowed Google Chrome users to be notified when MITM was being performed using the *.google.com certificates obtained through the DigiNotar hack [Pal11]. More recently, on December 24, the usage of a certificate from an unauthorized CA for '*.google.com' was detected and blocked by Google. The certificate was only used inside a company's network, called EGO. The rogue certificate at EGO was detected, because someone within the organization's network used the Chrome browser to access a Google site, where obviously, the received certificate did not match the Google's own pins [Ric12]. Therefore, we can say the static solution of Public Key Pinning is operational, enables faster detection, and can indeed decrease the risks.

Although the risks can be potentially lower using this solution, there are still issues to be solved before Pinning should and will be implemented in a large scale by all browsers. Adam Langley even wrote: "Although Chrome has Public Key Pinning for some domains, which limits the set of permitted certificates, we don't see Public Key Pinning as a long term solution (and nor was it ever designed to be)."[Lan11a].

The solution, if extended to work for all domains, will become a huge database that is managed by browser vendors. The interviewee, the assistant professor in the field of cryptography with whom I also discussed this subject, said that this kind of solution divides the risk into compartments, but that it is another database that can get contaminated, see Sec. A.4.

If there is something we have learned from the MD5 collisioning, which resulted in a valid certificate, see Sec. A.4, is that if processes are done automatically, they can be fooled; and if processes can be fooled, they can not be trusted. Thus, if Pinning is done in an automatic fashion, the system can not be trusted completely, resulting in a easily contaminated database. This brings us to another security issue of Public Key Pinning: the user's first visit.

Since the pins are kept offline, their creation of pins becomes critical. This happens when a user visits a domain for the very first time, which can be fooled by performing a MITM attack. The user will then not be able to distinguish whether the newly stored pin is correct, since the user has no previously stored pins for that domain. This problem is called the bootstrap problem and can be solved if all pins were preloaded into the browser. However, preloading all the pins is, according to the authors, would bring scalability problems along, which they are still trying to deal with [Pal11].

Pinning makes sure the user will, after the first visit, only trust the pinned CA that

is stored by the browser and get a warning when certificate from other CAs is presented. Lets suppose that an bootstrap attack has been executed successfully and the user has been fooled to pin a domain to a already compromised CA. Then the user will only accept certificates from that compromised CA and get a warning message when the true certificate is showed. This scenario would worsen the situation, since the attack becomes more permanent.

Another important issue is when a domain owner loses or loses control of their domain's private key. In this case, the domain owner will need to take multiple actions: revoke the previous key at the CA, revoke previously pinned SPKI's at all parties that support Pinning, create new public and private key pair, request a new certificate at a CA, and pin the new public key. The extra actions needed are only required when public keys are being changed. Deploying Public Key Pinning safely will require operational and organizational maturity [EP11] since domains may make themselves unavailable by the present of invalid SPKI pins. Domain owners will need to keep track of which pin is still valid.

The design of Public Key Pinning does not cover other applications of PKI. It will probably be possible to allow Public Key Pinning in operating systems as well, if operating system vendors would change their operating systems accordingly, however, this has not been discussed in the designs.

Another concern is the placement of responsibility at browser vendors and the business model of this solution. It is unclear whether domain owners will need to pay every party that will support Pinning or that parties will just add this extra load of work as support.

5.2.2. Solution aspects

This is a summary of the various aspects of Public Key Pinning, grouped by type, mentioned in the previous paragraph of the solution analyst, according to the aspects discussed in Sec. 1.3.1.

Security	Usability	Costs
Risks	**Efficiency**	**Transition costs**
- MITM attack can fool the user, when visiting a website for the first time, into thinking that another CA is the right CA for that domain. - Revoking a public key can take too long because multiple parties are involved. - Database at browser vendors, containing the pins, can get contaminated, either by mistakes or attacks, resulting in warning messages for valid domains. This also holds by the offline database in the user's browser.	- Since the solution will be implemented in software, there will be no latency - Slower acceptation time for a certificate is not avoidable with this solution, since an extra check on the issued certificates are needed. This should not be done automatically because if automatic, the hacker of a CA will then probably also find the function that adds the certificate to the database of browser companies.	- The browser and operating system parties will need to change their software in order to remember and test for Pinning. - Domain owners need to allow Pinning by adding support for HSTS into their code.
Improvements	**Control**	**Maintenance costs**
This solution is able to limit the risks and powers of CAs that are compromised. Warning messages will be shown to users when a certificate is presented by another CA than the pinned CA.	The users will still have the same amount of control for deciding which Root CAs to trust. It is however not clear how much control the user will have over the pins.	- Browser vendors will need to maintain their database of 'pins', which can be expected to bring along much costs when the solution is deployed for all secure domains. - Domain owners will need to keep checking if the correct pin is set for their domain.
Sustainability	**Availability**	**Educational costs**
- The solution is dependent on SPKI technology. - Internet will be dependent on browser vendors.	- Vendors supporting Pinning will need to change their software and acquire new hardware for storing the pins. Already available software will also be used. - Invalid pins will make a domain unavailable.	- The code domain owners will need to add to their domain is relatively simple and will not take much time. - Browser vendors supporting Pinning on the other hand, will need to take great measurements to allow Pinning and its management.
Juridical	**Limits**	**Losses and profits**
Browser vendors, supporting Pinning, will be responsible for taking care of their pin database. They need to have a proper system for revoking pins that are no longer valid.	- The solution does not obligate domains to include pins in their HTST header. Similarly, private domains, like secret.foo.com, will definitely not add Pinning. - The design will experience scalability problems for preloading the pins into the browsers. - The design currently only covers webbrowsers.	Browser vendors will need to invest much time and effort to make the Public Key Pinning available. Also much organizational and operational costs will be involved, since the database with pins must stay valid.

5.3. Perspectives & Convergence

The main idea of the Perspectives Project at Carnegie Mellon University from 2008 [WAP08] was to decentralize the trust to allow secure communication and to prevent MITM attacks. The solution has some similarities to the decentralized PKI in the "web-of-trust" model used by Pretty-Good-Privacy (PGP) [CDFT98], however PGP's primarily challenge is to estimate the strenght of key trust chains, each representing a pair of real world acquaintances, because it uses human contact to bind entities to Key. Perspectives does not have trust chain, but notary servers that keep an eye on each other.

The basics of Perspectives, much alike the basics of the current CA system, is that a client asks a third party's opinion in order to make sure the certificate the desired network service has given, truly belongs to that service. The main differences are that in Perspectives, after the client has received the certificate of for example an SSL site, the client can ask more than one third parties to check the site's certificate. If all or at least a certain percentage of the third parties see the same certificate for the requested site, then the client can trust that there is no MITM involved and the certificate truly belongs to that website, see Fig. 5.1.

Perspectives' third parties, called *notary authorities*, are independent from the currently used third parties, the Certificate Authorities. A notary authority is responsible for a group of *notary servers* and determines which machines are legitimate notary servers. A notary server can be added to the system by sending its public key and IP to a notary authority. Notary servers are coordinated groups of hosts, that are distributed across the Internet and keep records of their answer to client's queries. A notary server provides clients with an application-independent query interface and uses application-aware *probing modules* to monitor different types of services (e.g., SSH or HTTPS). The probing module observes keys by connecting to the service and mimicking an ordinary client until it receives the service's public key, at which point it disconnects. Each notary server uses a local database to store a service entry for each monitored service from some server. A service entry contains all observed key data the notary has recorded and their timespans, which indicates a period of time during which the notary observed that key. When the notary makes a new observation it adds either a new key or updates the key's timespan.

According to the design, notary authorities publish the list of notary server's IP addresses and public key pairs to each notary server in the system. The list is then signed by the notary authority, which can be verified by notary clients when they download the list from a notary server. Notary clients are integrated into applications, such as browser software, and will have the public key of each notary authority using an out-of-band mechanism, such as Tor [WAP08]. A client will try

to contact a notary server whenever a key from the service the client tries to contact to does not match an existing entry in the client's cache. This may occur because the client has never contacted the service before or because the previously verified certificate for a service has changed or expired.

In order to decide whether to accept or reject a key that is offered by the server the client wants to connect to, the client's *key-trust policy* determines n notary servers for contacting. The client then randomly chooses n entries from the list of known, trusted notary servers and queries them in parallel using UDP. The query process is completed once more than q notary servers have replied. Here, q is defined by the key-trust policy. A key, flagged as the correct key by q notaries, is called quorum. The duration for which a key has been in quorum, is called quorum duration and referred to by d.

The final aspect of the notary design is data redundancy, which is a cross validation mechanism, allowing the detection and limitation of compromised notary servers. This mechanism requires each notary server to act as a shadow server of several other notary servers, which verifies the observation history of the notary they shadow. This prevents compromised notaries to change their observation history and to pretend as if a key has been seen for a longer period of time. A notary's observations are considered valid by a client, when the notary has at least r number of shadow servers confirming the notary's history.

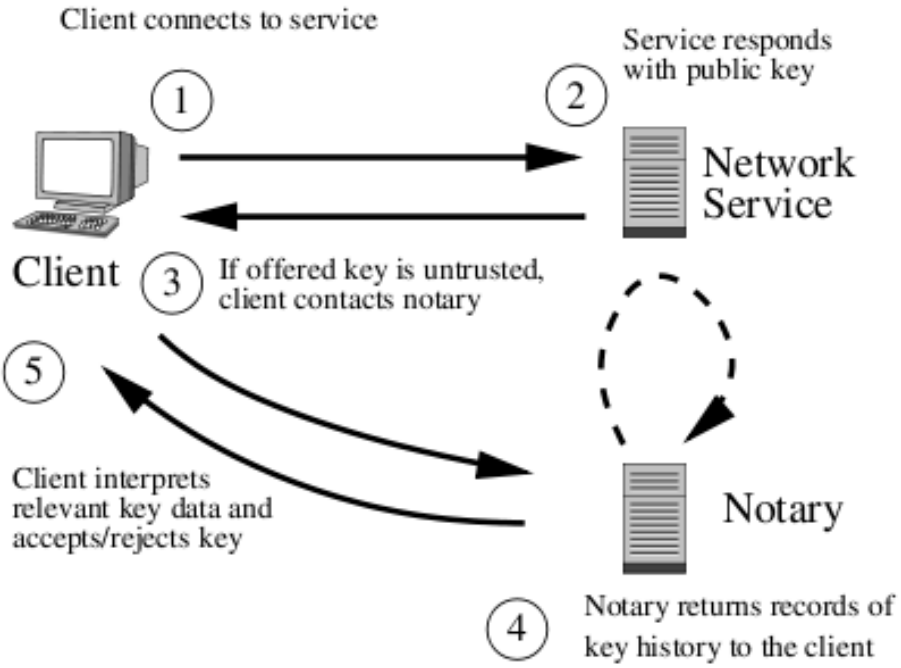

Figure 5.1.: Principal of Perspectives

The values for n, q, d, and r can be specified by the user in order to specify their

desired level of security. Higher values for these variables increase the chance the certificate truly belongs to that server and decreases the chance to a MITM attack. However, if these variables are set to too high values, then the risk of denying valid certificates exists. A high value for q, for example, can deny a certificate if some notaries are unavailable or compromised. Similarly, a high value for d can result into denying newly changed certificates. The queries returned by the notary servers is a key history for the service of the requested server.

Although the design of Perspectives shows a self controlled mechanism, it has also some serious issues that, if not fixed, will prevent it from becoming a replacement for the current CA system. The solution is available for users as a Firefox add-on with notary servers hosted by the Carnegie Mellon University Perspectives team.

The solution is described by the authors themselves as "Not bullet-proof, but provides a security trade-off suitable for many non-critical websites." [Wen08]. Therefore, Perspectives is not seen as a standalone solution for PKI by itself in this thesis, but rather as the underlying basis of another solution, presented by Moxie Marlinspike in 2011. The solution, called Convergence, was presented at the Black Hat conference in Las Vegas [Mar11a], where Moxie Marlinspike showed during his talk titled "SSL And The Future Of Authenticity" the improved Perspectives.

Convergence, which is built upon the Perpectives' design, improves some major issues such as completeness, privacy, and responsiveness. Besides the technical improvements of Perspectives in Convergence, a main difference between the two is the active participation of users in Convergence. The implementation, available as a plug in for Mozilla's Firefox, allows users to control which notary servers to trust or distrust. Putting the "Trust Agility" in the hands of the users and stepping off the current Internet Certificate Authorities system, is also how Moxie presented Convergence. The solution was motivated by the fact that the current system does not allow users to distrust a CA without denying themselves the access to certain websites.

The first issue, completeness, solved by Convergence is about checking all HTTPS requests. In Perspectives, only the initial connection is validated by notary servers, where background content, such as images, Javascript, and CSS, is not validated. In Convergence, the validation is performed for all connections.

The second issue in Perspectives, addressed by Convergence, is the privacy of users. Although users randomly choose n number of notary servers to query, each of these notary servers will know the connected user's IP addresses and the services they wanted to connect to. Since anybody is able to set up a notary server, this is a huge privacy issue. This is resolved in Convergence by making use of bounce notaries. A bounce notary is a by the client randomly selected notary server, assigned to become the bounce notary for the client's query. The bounce notary will then serve in the same way ISPs will function in DNSSEC, as an in-between server that is only aware of the client's IP and destination, but not aware of the content. This is carried out through the usage of public key cryptography, where each notary has its own public

and private key pair. Because the public key of each trusted notary is known to the user, the user is able to encrypt the request for each notary to the bounce notary. The bounce notary will send each encrypted request to the destined notary server and send their answer's back to the client. Using this method, the IP of users is not revealed to the notary servers, as they only receive the desired service of clients. However, the user's privacy can still be invaded when two or more compromised notary servers would collaborate.

In Perspectives, notary servers check the certificate of the desired service for every query and make no use of caching, resulting into significant notary lag. In Convergence, this is solved by local caching at clients and caching at notary servers. In the new design, observation of clients is sent to the notary server the client queries. When a notary server is queried, the notary server will check the client's observation against its local cache and in case of mismatch or missing certificate, will check the certificate of the desired service. To keep the local cache of notary servers up to date, Moxie said, they will pull certificates from the services once a day. Successful observations are also saved in the client's local cache, which reduces the number of queries and eliminates the notary lag when the client receives a service's certificate that matches the client's local cache.

Another improvement of Convergence is that notaries are allowed to have a different number of back-ends. By default, this is set to the notary network Perspectives, but notary servers can for example also make use of DNSSEC or the current CA system, making the notary servers extensible to new technologies.

5.3.1. Analysis

Establishing trust without the need of the CA system and at the same time preventing the MITM attack would definitely be a great idea. With Convergence, a solution has been made available where users are able to take control over the trust by only trusting the notaries the user itself finds trustworthy. The solution also takes the trust responsibility away from browsers, where browsers are not needed to have responsibility over which root CAs to trust anymore. This can be very handy for users, since a browser can decide to keep their trust in a CA due to other reasons than user's safety, as happened with IE during the DigiNotar compromisation.

Convergence replaces Certificate Authorities with notary servers which can be created and managed by anyone, which is the greatest disadvantage of this solution, not allowing it to be applied in large scale. As the interviewee in Sec. A.2 pointed out that governments and large companies would not appreciate their security and data to be dependent on unknown parties. For them, there is much at stake and attackers, possibly from other governments or large companies, are willing to spend much time and effort to prepare an attack. This was confirmed by the interviewees in Sec. A.2 and Sec. A.4, where they both mentioned that rather big organizations or governments can not rely their trust on notaries which they don't know for sure

can be trusted. They will need an official authority that guarantees the state of the notaries.

The fact that a self signed certificate is enough to establish trust when the user's notary servers agree they see the same certificate for a service, can be seen as a great advantage to use solution, because it removes the costs to acquire certificates from Certificate Authorities [Pag11]. Services can then provide encrypted communication between their clients, without the need to buy an certificate. However, this property also makes it easier to eavesdrop users, since the only barrier attackers need to overcome is to create as many compromised notary servers as they can and lead users toward services that pretend to have a certificate of a service they do not actually have. With this property, large attacks will be able to mimic any website. They could for example pretend to have an EV certificate of a large bank and intercept all users connection.

Similarly, the idea of the solution will definitely work for people that are already in a safe area, with some simple attackers that try to perform MITM attack. However, large organizations and governments will still be able to eavesdrop users, since they are capable of setting up many compromised notary servers. For governments and ISP organizations, eavesdropping becomes even easier with this method, since they can either get access or have control over the ISPs. This allows them to perform MITM attack for many users simultaneously by redirecting clients toward the compromised notary servers. Setting up a notary server is a fairly easy task to do and can be performed on any Linux computer [Con]. As Moxie Marlinspike mentioned [Mar11b] during his presentation: *"Notary implementation is available free and open source. Anybody can run their own notary. It requires very little resources and is designed to be extensible"*.

This brings us to another problem, which is the question of who is eventually going to run and maintain the notary servers. It is currently assumed that notary server can be hosted by anyone, thus either companies or users, but this is not something we would like the complete Internet be dependent on. If users are going to be the one hosting the notary servers, then it can not be guaranteed that they will be willing to keep hosting notary servers for a very long time, since this will cost them at least some bandwidth and processor time. However, if the notary servers are going to be hosted by companies, then how are we going to be sure they will not abuse the data on the notary servers. As mentioned before, only two notary servers need to work together to avoid the privacy solution in Convergence. A good point mentioned by Adam Langley in the same post [Lan11f] is: "*Also, we have a very strong interest for the notaries to function, otherwise Chrome stops working. Combined, that means that Google would end up running the notaries. So the design boils down to Chrome phoning home for certificate validation. That has both unacceptable privacy implications and very high uptime requirements on the notary service.*".

Another property of Convergence which is seen as an advantage is the ability of users

to control which notaries they trust. However, this can also be seen as a weakness, since most of the users do not understand security and would not change the default settings. As Adam Langley pointed out [Lan11f]: *"Although the idea of trust agility is great, 99.99% of Chrome users would never change the default settings. (The percentage is not an exaggeration.) Indeed, I don't believe that an option for setting custom notaries would even meet the standards for inclusion in the preferences UI. Given that essentially the whole population of Chrome users would use the default notary settings, those notaries will get a large amount of traffic".* According to this, the default set of notaries that are included in the client software, will stay the only notaries many clients will trust. This will require extreme high traffic bandwidth from these notary servers and will make them very attractive for attackers, which will be definitely easier to hack into than most CA servers.

5.3.2. Solution aspects

This is a summary of the various aspects of Perspectives & Convergence, grouped by type, mentioned in the previous paragraph of the solution analyst, according to the aspects discussed in Sec. 1.3.1.

Security	Usability	Costs
Risks	**Efficiency**	**Transition costs**
- The solution is dependent on ISPs, where an compromised ISP can divert clients to compromised notary servers. - Attacks become easier since there will be no need to bypass an authority; only the need to set up many notary servers. - Browser vendors can endup being responsible for the notary servers, resulting into unacceptable privacy implications - Users have too much control. Most users will not use them, however, the settings must be protected very well.	- The extra bounce notary and notary servers that need to check a domain certificate can result into connection latency, though not always, since the solution uses caching.	- Browser software and Operating Systems needs to be changed. - Many new notary servers needs to be created and hosted.
Improvements	**Control**	**Maintenance costs**
The trust is decentralized where users themselves can choose which notary servers they can trust. Also, the ability to trust self signed certificates, without the trust of a CA is made possible	The user will have full control over which notary servers to trust, which is not a good approach for average Internet users.	Notary servers must be maintained and users should maintain their own trusted notary servers.

Sustainability	**Availability**	**Educational costs**
Notaries can use different protocols, such as perspectives, current CA system or DNSSEC, to check the certificate of a service.	- Convergence does not need the software of the current CA system. There is the need for setting up notary servers, notary authorities, notary clients, notary shadow servers, perspective software, and changing browser software. - The availability of more notary servers are needed	Any educational cost would be for browser and Operating System vendors that would need to change their software.
Juridical	**Limits**	**Losses and profits**
There is not a real person that can be held responsible if something would go wrong.	- The solution is currently used to check website certificates, which leaves out other PKI solutions such as code signing. For now, the solution can only be used for websites through an Firefox plugin. - The solution will not be used by governments and large companies, since there is no authority.	Self signed certificates are allowed, removing the obligation to request a certificate from a Certificate Authority. CAs will definitely make some losses when this becomes popular.

5.4. Sovereign Keys

Sovereign keys solution is an EFF's project that allows clients to validate a domain's public key by its sovereign key, without the need to trust any third party CA. The solution keeps an history of sovereign keys for domain services in an read- and append-only data structure, which ensures that previous claims about the domain's sovereign key cannot be changed or denied. The sovereign keys are basically another public and private key pair which are created by domain owners for their domain or a particular service of their domain.

Timeline servers (TLS) are responsible for issuing and maintaining the sovereign keys. Each timeline server has its own public and private key pair, which is used to sign a sovereign public key and to authenticate themselves. The public key of each timeline server is known in user's client software and there will be approximately a set of 10 to 30 timeline Servers [Eck11d]. After signing a new sovereign key and adding it to the timeline server's data structure, the timeline servers will synchronize the registration with the other timeline servers, in order to make the claim for the domain's sovereign key read- and append-only. A TLS is said to be approximately 2 to 3 TB of disk storage.

For verification, but also for performance and privacy purposes, the trusted timeline servers are replicated and kept up to date on multiple **mirrors** around the world

[Eck11f]. Users can query any of these mirrors in order to verify a domain's public key against its, by timelines registered, sovereign key. When using sovereign keys, users will not need to trust in a CA anymore.

Domain owners can register a created sovereign key at a timeline server by identifying themselves using some form of entity-dns association. Identification can be done by either the current CA system or the future DNSSEC, which is explained in Sec. 5.6. This bootstrapping from the existent solution is necessary to prevent the same bootstrap problems that was experienced in Public Key Pinning, see Sec. 5.2, where the uncertainty of user's first visit existed.

It is like using the Internet Service Providers with secure connection, where the ISP itself is not trusted, but relied on for its service. The ISP can send the packets to the desired destination, but is also able to send them to a wrong destination. ISPs are able to deny the service, but cannot deceive you. Similarly, Sovereign Keys uses the service of the existing CA system or DNSSEC in the future, to provide a secure initialisation [Eck11d]. However, similar to the ISP example, the existing system could undermine the trust and cause Denial of Service when for example the responsible CA would get compromised. Thus, the bootstrapping technology, such as CA system or DNSSEC, secures the relation between public keys and entities, where sovereign keys secures the relation between domains and a public keys. Before accepting a sovereign key by a domain owner, its X.509 certificate or DNSSEC response is validated by the issuing timeline server.

Sovereign Keys were presented at the 28th Chaos Communication Congress in December 2011 [Eck11e] by Peter Eckersley as a proposal to fix the attacks such as the MITM and server-impersonation attacks, to allow faster detection of fraudulently issued certificates by compromised CAs, and to provide a workaround when an attack would trigger a certificate warning message that users mostly ignore [SEA+09]. The workgroup is still working on the method and has presented the work in order to get feed back for further improvements.

Sovereign Keys can be created for a particular service under a particular domain by its domain owner. The system can be used upon the existing PKI without complications until it is fully integrated over time. Though, clients that understand Sovereign Keys are ought to verify the public keys for a particular service, e.g. HTTPS or SMTPS, on a particular domain that has already a registered Sovereign Key. If the server's public key cannot be verified by the corresponding Sovereign Key, the client must refuse to connect to that server. Similarly, a client must refuse to connect over an insecure protocol, such as HTTP or SMTP, when a Sovereign Key exists for that domain, because the mere existence of a Sovereign Key states that the domain uses secure protocol.

Each new sovereign key entry in a TLS includes a strictly-incrementing serial number and a monotonically-increasing timestamp, where the entries are cryptographically signed by the TLS's private key. The entry also consists of the protocol, port, expiration date, and a wildcard. The 'protocols' field is a list of strings, such as

HTTPS, SMPTS or IMAPS, and denote the services that must be signed by the sovereign key on that domain. A service may specify a specific port number to specify their preferred routes for attack circumvention. Sovereign keys also have an expiration date, after which they will become invalid [Eck11d]. The wildcard field indicates whether or not this sovereign key is valid for any subdomain. The protocol also has an 'In case of revocation' containing the field 'inheriting name(s)', which determines which CAs, other than the one noted in the entry, can be used in case this one is compromised and subsequently revoked. The list can contain zero or more names of CAs, such as 'Verisign.com'.

The level of trust needed for timeline servers is very low, because the sovereign key protocol is able to cryptographically verify the important functions they perform and is able to revoke the trust for renegated timeline servers. Mirrors increase the availability and performance from the clients' perspective. Mirrors do not record any new events to the timeline servers, but only synchronize with the authoritative timeline servers, verify timeline integrity and answer to client queries. A mirror can also observe, identify reneged timeline servers and add them to a structured list which can be queried by clients. On each mirror, at least two different timeline servers are replicated [Wie]. Mirrors are identified by an IP address, a port number and a public key. When a client asks for a name, a mirrors responds with an *entry since*, a *freshness message*, and the mirror's *signature*. The *entry since* is the serial number of the sovereign key. The freshness message is a time stamp that indicates when the last update for this sovereign key has been given. [Eck11a]

In the event that a timeline server contains a fault, such as two events being recorded out-of-sequence, any client can notice this and flag the server as renegade, meaning that trust in the server will be revoked. To track these servers, *renegation tracking* field is added in the protocol, which is a 32 bit number of a hash of previously trusted timeline servers the mirror or client knows to have been reneged. If this differs, a synchronization will be held where at the end of the process, both participants should have the same list of reneged timeliness.

To circumvent and protect the user against MITM attack, the Sovereign Key Specification draft noted that Tor's hidden services could be used as a fallback mechanism for users who are affected by MITM or connection blocking attacks. On particularly hostile networks, it is presumed that these users would access the Tor network via Tor bridges or similar tunneling mechanisms.

The power of Sovereign Keys is its ability to be verifiable by clients, domain holders, mirrors, and other timeline servers. Nonetheless, it is stated in the draft design document that a governance mechanism may be required for multiple purposes such as monitoring timeline servers' availability and their ability to push updates to mirrors, maintaining a master list of all of the timeline servers that have ever been trusted or to maintain a list of TLDs and CAs. It has not yet been decided, but the timeline servers could be operated by different entities like the EFF itself, or Mozilla, Google or Microsoft [Tew12].

The project is currently being implemented, and the code, as far as it is for now, can be reviewed at the website of the project [SKS].

5.4.1. Analysis

The extra layer of security Sovereign Key is offering would make it harder for an attacker to abuse PKI, since besides creating a fraudulent certificate, the attacker also needs to add or change the Sovereign Keys for that domain. The extra layer also will remove the need to trust third parties from the existing PKI after bootstrapping from it and decentralizes this trust.

Although the Sovereign keys proposal sounds intriguing, at first, the solution may also sound like it tries to replaces one problem by another problem through the addition of timeline servers. Many people feel this way [Sch11] and think that building a solution on an already insecure solution is asking for problems. However, the working group recognizes many of the problems and is searching for solutions. In the 'issues' directory of the Sovereign Key's Git repository, some technical and practical problems with the proposal is discussed [SKS]. If capable of solving the issues, Sovereign Keys can decrease the risks and improve the discovery time for compromised CAs or fraudulently issued certificates. The design also avoids the need to show the confusing certificate warnings, which users will often click through even when they are under attack [SEA+09].

The first issue mentioned by Eckersley [Eck11g] involves the transition between the CA system and Sovereign Keys. The creation of Sovereign Keys, in theory, could also be performed by an attacker. To perform this, the attacker would need to compromise the victim's DNS or web server to obtain or create certified private keys and create a Sovereign Key accordingly. If the creation of a fraudulent Sovereign Key would occur successfully, the trust in the fraudulent certificate will be even more or even permanent. The authors have come up with some solutions to make it more difficult for an attacker to create an fraudulent Sovereign Key. Suggested solutions are for example sending multiple emails listed in the WHOIS of the domain and the certificate containing a confirmation and cancellation link. The authors mention that is it an empirical question whether these methods are sufficient to make successful sovereign key creation attacks rare. Although obtaining a private key by an attacker is something the domain holder should pay attention to, it is a Sovereign Key issue that fraudulently created Sovereign Keys can, in worst case, give permanent trust for that key.

Alongside the creation of Sovereign Keys, bootstrapping mirrors seems to be another issue. Ben Laurie, from Google, said that the issue in SK is that the relying party needs to trust mirrors to tell it what the current status of any particular domain is (i.e. what the current key is), because the only other way to be sure is to download the entire database, which will be many gigabytes long [Lau12]. It is expected that the mirrors receive their first data using data distribution mechanism such as Bit

Torrent or through the exchange of a hard drive [Wie], which would only require the mirror to update.

However, even if the data distribution would work, an organized attack, can launch large amount of compromised mirrors and fool the clients, which is also called the Sybil attack [SKs11]. Furthermore, these mirrors learn the user's IP address along with the information requests to visit a domain. The problem is the same as with Perspectives & Convergence, since they are both using third parties as a trust anchor.

More important issue is the assumption that the set of authoritative timeline servers is small. The number of timeline servers is stated to be between 10 and 30, which is a small number in order to get away from the decentralized model that is suggested and sets the focus of the attacks towards the timeline servers. The authors wrote [Eck11b] that: *"It is an open question whether we need to support greater degrees of decentralization."*

The design document has also left out details regarding the organization of the timeline servers and mirrors. It is for instance not yet clear who will pay and manage the timeline servers and mirrors or will be held responsible if an attack is made possible through one of the hosted timeline servers or mirrors. The decision of who eventually will be hosting the timeliness is critical. If it is not going to be authorities, the large companies and governments will not want to use Sovereign Keys. However, the solution must also be applicable for them, because otherwise cheaper and easier solutions can be used to achieve the same effect for normal users, such as PGP or the solution given by Moxie Marlinspike, see Sec. 5.3. As the interviewee in Sec. A.2 explained, for large companies and government, it is preferred if the entities we should trust, are hosted by some organization that can be held responsible. On the other hand, if the timeline servers are going to be hosted and maintained by authorities, the protocol must make sure the synchronization problems are solved perfectly. Otherwise, Sovereign Keys will have no extra value to the existing CA model when a country decides to deceive their users.

Another problem is the way timeline servers are supposed to be trusted and managed by users. Embedded keys of timeline servers in clients is suggested, but this may lead to update issues as timeline servers are reneged, changed or added. This is an important issue, since the extra layer of trust will then not only add no extra trust, but will only cost a lot of money and time, since many changes must be applied. According to Moxie Marlinspike, the creator of Perspectives and Convergence discussed in Sec. 5.3, sovereign keys would require a major Internet migration, changing both the way that every web-server deploys SSL today, as well as the way that every SSL client processes server certificates [Con11].

Increased latency is yet another concern, since there are extra queries that mirrors must perform. The authors wrote [Wie, Eck11d], however, that the latency of the Sovereign Key query is less than or equal to the latency of establishing the TLS session in the first place, and that it will add no further latency.

Since the sovereign keys design allows domain operators to reduce the number of

third parties who can launch attacks against their services to zero, it has the property that if domain holders would lose their sovereign private key, they will probably lose the ability to switch to new operational keys, or may even lose control of their domain, until the Sovereign Key expires. Prosposed measurements for this is to either rely on a service provider the domain holder would trust or the creation of sufficient amount of backups by the domain holder itself [Eck11d].

A mystery to me is why the protocol is considering to implement 'unbind', which would re-establishes an earlier revoked Sovereign Key [Wie]. It seems logic that a revoked key would have been revoked for a good reason. Un-revoking does not seem to be an ideal functionality.

5.4.2. Solution aspects

This is a summary of the various aspects of Sovereign Keys, grouped by type, mentioned in the previous paragraph of the solution analyst, according to the aspects discussed in Sec. 1.3.1.

Security	**Usability**	**Costs**
Risks	**Efficiency**	**Transition costs**
- Timeline servers can become the new prime targets for attackers. DoS attacks can be performed by flooding the timeline server with registration requests and mirrors by query requests. - Privacy issue occurs because all users, for all websites, need to visit the mirrors. Although, there are many mirrors where visitors can choose from. - Concerns about the trustworthiness of timeline servers and mirrors. Mirrors are said to be not very expensive and can be set up relatively easily, which increases the chance to compromised mirrors. Who and how timeline servers are managed, is not yet clear. - Update and synchronization problems which can, in worst case, lead to permanent trust of entities that should not be trusted. - Key loss of domain holders can result in loosing the control over their domain.	Although the authors noted that there will be no difference with latency, this must yet be proven, because there are many extra queries and updates that are required to ensure the system is trusted.	New timeline servers and mirrors are needed to be set up. The mirrors are said to cost around 100 dollars. Browser vendors will need to change their software in order to support Sovereign Keys. Software, created by companies, will also need to change if they want to collaborate with Sovereign Keys.

Improvements	**Control**	**Maintenance costs**
- The design is such that it can recognize automatically which timeline servers or mirrors are compromised, revoke their trust if necessary, and help identifying compromised CAs. - The Internet will work when it is safe, and else it will not work. - No warnings, which users don't use, are needed anymore.	- Users will have no control over which timeline server to trust, which can be discussed to be a good or bad thing. Their clients, however, can revoke the trust in a mirror if it does not perform according to the protocol, but still, this is not managed by the user. - From the point of view that users learn to ignore the warning messages anyway, Sovereign Key does not show any warning message, but only a not working page.	- The timeline servers and mirrors will need maintenance to stay operational. - Domain owners or intermediate parties must maintain the sovereign private key, since loosing the key and all backups will result into loosing control over the domain.
Sustainability	**Availability**	**Educational costs**
The solution is said to be able to work with the current CA model and the newer techniques, such as DNSSEC, to bootstrap from.	- New changes are required, not only to web browsers, but also to software developed by companies, to authenticate using Sovereign Keys. The changes do not need to happen immediately, since Sovereign Keys can be implemented while working with the current CA system. - Also important is that if attackers would flood the timeline server or the mirrors with requests, creating a Denial of Service attack, resulting in unavailability of the Internet for users of the corresponding timeline server or mirror. - Loosing the sovereign private key will result into loosing control over the domain until the sovereign key is expired.	There will be no profit for anyone when this solution is added to the system, only losses: - Browser vendors will need to invest money and time to implement this solutions. - The entity that will host and maintain timeline servers or mirror, will also need to invest money and time. - Companies will need to change their software if they want to work with Sovereign Keys

Juridical	Limits	Losses and profits
Governments and large organizations will probably be still able to prosecute their wanting. Even if the timeline servers are hosted by third parties, they could set up large amount of compromised mirrors. New power is given to the hosts of the timeliness and mirrors, however users do not always have to connect to the same mirror.	Since the user will not get a warning message anymore, but just a not working page. Also, the user will have no control over which timeline server to trust.	There will be no profit for anyone when this solution is added to the system, only losses: - Browser vendors will need to invest money and time to implement this solutions. - The entity that will host and maintain timeline servers or mirror, will also need to invest money and time. - Companies will need to change their software if they want to work with Sovereign Keys. - Domain owners will need to create a sovereign key and maintain thei sovereign private key securely.

5.5. Certificate Transparency (CT)

> A very short explanation for Certificate transparency by Ben Laurie: *"Certificates are registered in a public audit log. Servers present proofs that their certificate is registered, along with the certificate itself. Clients check these proofs and domain owners monitor the logs." [Lan11c]*

Certificate Transparency was first introduced in November 2011 by Adam Langley from Google, explaining the concept of Certificate Transparency in his blog ImperialViolet [Lan11a]. The idea was inspired by conversations with EFF about the Sovereign Key Project [LL11], see Sec. 5.4. In Sovereign Keys, the main goal was to allow faster detection of CAs that have been compromised by keeping a history of their issued certificates in timeline servers. Certificate Transparency goes one step further and allows certificates to be seen as valid, only if they are published publicly in an append only list. This allows detection of fraudulently issued certificates even before they are used. The idea is that domain owners keep an eye on the issued certificates for their domains and revoke any certificate that has not been requested by them.

The design of current Public Key Infrastructure is such that Certificate Authorities do not have to make their issued certificates publicly known. This means that trusted Certificate Authorities, compromised or not, are capable of issuing certificates for any domain and use them without anyone being notified until they are used and noticed by users. Since these certificates are seen as valid certificates, users or browsers are not able to notice the difference and proceed the communication as

normal. Thus, private schemes do not protect users from CAs acting in bad faith or CAs which have been compromised [Lan11b].

The idea is that when certificates are publicly known and searchable, browsers, domain holders and users can collaborate to ensure the log is honest. Domain holders and other interested parties can monitor if there has been any fraudulently issued certificate for their domain. Thus, domain owners can verify if only their own legitimate certificates are in circulation, and can take action when other certificates for their domain are founded [LK12].

Adam Langley, Ben Laurie and Emilia Kasper have been working on this solution since then. Their idea is that every certificate will be logged in an append only audit log that is managed by browsers. Every log entry of a certificate is accompanied with a signature of the CA issuing that certificate, which is called the audit proof. The log is build using a Merkle tree, which can, according to the authors, be built efficiently [CTwa]. The Merkle tree is a type of hash tree, where each parent in the tree is the concatenation of the hash of its children. In the Merkle tree, only the root needs to be trusted in order to trust the rest of the tree. However, to validate a certificate's hash, the user must have more parts of the tree than only the root [CTwb].

The structure of the log is such that it is possible to provide proofs of consistency using snapshots, which is the state of the log at a particular time. Inconsistencies indicate dishonesty on the part of the log. It is possible to prove the consistency by testing two versions of the tree's snapshots. Two snapshots are consistent if the later version includes everything in the earlier version, in the same order, and all new entries come after all entries from the earlier version, which is the same as saying the log is append only. The size of this proof is logarithmically proportional to the number of entries .

In the event that the snapshot of a certificate's log is in the future, with respect to the latest audited snapshot by the user, or in the case that fetching consistency proofs snapshots is unavailable, then the user has to record the snapshot and try again to validate it later. Failure to validate the log within a certain amount of time, known as the Maximum Merge Delay (MMD), is considered a breach of contract by the log. The client can then start the process of reporting misbehavior by the log to the browser vendor. In this solution, the browser vendor is going to be both the auditor and the party to which the clients will report suspected problems [CTwa]. If the MMD would not be set, then adding a certificate entry to the log could be delayed and used to attack users without detecting this [LK12]. Thus, the MMD is the longest possible time a rogue certificate can be used without detection.

It is important to note that logs will not deal with revocation. It has been suggested that revocation of certificates would be accomplished by existing mechanisms or the Revocation Transparency (RT). RT is similar to CT, containing a list of revoked certificates in a Merkle tree, except that RT needs a recent status for the certificate and CT does not.

5.5.1. Analysis

Changing the Internet Public Key Infrastructure such that everybody can monitor the certificates issued by a CA, is recognized to be a good idea and the BoF (Birds of Feather) even determined that CT should proceed along without having a working group [Mor12b].

The solution, however, is not yet in a advanced stadium where there are still many aspects that need to be defined and elaborated. One of these aspects is how the logs are published. It is not yet very clear if each CA should have its own log or that the logs are centralized and managed by browser vendors. It is yet to be decided who will be responsible for the logs, but seems to be the browser vendors. Making browser vendors responsible for the logs, however, can bring many administrative problems along in order to make an issued certificate seen as valid by the users. Currently, certificate requests, at least by known clients, can be issued and used within hours, see Sec. A.1. If certificates can only be used after they have been published on the log managed by browser vendors, then the time between the request for a certificate and the usage will definitely increase. Also, the log can encounter scaling issues, because of the size, and will become a single point of failure, which will be very attractive for attackers. The alternative, where the logs are managed by CAs themselves, requires some kind of mechanism, making sure the CAs are not capable of issuing a fraudulent certificate and not add it to the log.

Another aspect that yet needs to be defined is how certificates will be revoked. It has been suggested that a separate audit proof, containing the list of revoked certificates, should be created. However, similar to the previous aspect, it is not clear who will be responsible for these lists. If the logs are managed by the browser vendors, there should be some system that ensures fast awareness and revocation, passed on by the CA in question. The time for revocation, however, will definitely be longer than when the CA itself would manage the revocation list.

The solution places more responsibility at domain owners, who will need to monitor the publicly available lists of certificates, in order to verify that there have not been certificates issued without their knowledge. The scenario exists that a compromised CA issues a certificate and adds this certificate to the log, which can then be used by the attacker as a valid certificate until its detection. Monitoring the logs for mis-issued certificates is therefore a critical part of this solution. It should be relatively easy to perform this monitoring for domain holders. Also, it is important to take in consideration how to educate domain holders in order to perform this monitoring fast and easy or otherwise, many small businesses that do not have the knowledge or the extra time, will suffer from this.

This solution, if attention is paid to the critical parts that yet need to be solved or worked out, will solve many of the trust problems in the current PKI: CAs will no longer be able to issue fraudulent certificates, either by government pressure or by compromisation, if they want to keep the trust of the rest of the world. Users will

be able to know for sure that they can trust a CA and the certificate for a domain. Not only because CAs will be more faithful due to the fact that using this solution everybody will be able to watch their issued certificates, but also because domain holders have a reputation to keep, and will therefore monitor and revoke any other certificate for their domain than the rightful certificates.

The critical points mentioned above, which are needed to achieve this great result, are not easy to solve. How the certificates are logged, revoked and by whom they are managed, are important issues. The response time for a revoked certificate is crucial in order to keep users safe. It is also important to determine what the consequences are if an issued certificate is not logged. Will users only get a warning or will the website be unreachable. If unreachable, a side effect of this decision is that self signed certificates will no longer work due to the obvious fact that self signed certificates are not trusted by a Certificate Authority and therefore not mentioned in any log.

On the other hand, self signed certificates are not supposed to be trusted in a CA system where we rely our trust in third parties. In this system, self signed certificates can not be trusted since a man in the middle attack could easily be performed, where the attacker sends a certificate of his own to the user, containing the attacker's public key. Browser vendors could solve this by giving the users the option to have a list of certificates that do not have to be checked with the log system. However, this could result in some serious malware security issues, which would also be able to add their own self signed certificate to such list.

Another small issue, which has nothing to do with security, is that by having the certificates issued by CAs made public, CAs will be able to have insight in their competitor's clients. CAs, or any other party, will be able to see how many clients and which clients a CA has. This information could be used to steal clients of a CA by other CAs or could be used by attackers for information gathering.

5.5.2. Solution aspects

This is a summary of the various aspects of Certificate Transparency, grouped by type, mentioned in the previous paragraph of the solution analyst, according to the aspects discussed in Sec. 1.3.1.

Security	Usability	Costs
Risks	**Efficiency**	**Transition costs**
- The logs can get manipulated, domain holders need to pay constant attention and make sure the trust in a log is revoke as soon as possible. - CAs could choose to not to add a certificate to a log - If the monitoring is not performed well by the domain holder, a fraudulent certificate can be trusted until detection. - A possible new vulnerability could also be hidden in the usage of Merkle trees. - Allows easy information gathering.	According to the authors, the log, using Merkle tree, can be built efficiently and checked efficiently. However, this still needs to be confirmed by experiments. Verifying a certificate will however take a little bit more time than the current system because of the addition validation with the Merkle tree, however this can probably be done very fast.	The solution would be accomplished without changing server software and not relying on third parties who would need up time and the ability to be reached. End user software needs to be changed.
Improvements	**Control**	**Maintenance costs**
- Issued certificates will be publicly available which can be used by domain holders to monitor issued certificates for their domains. - Government influence to issue a certificate will be visible to the world and noticed much easier and faster. - The trust in CAs issuing fraudulent certificates, either by compromisation or government influence, can be revoked. - Evidence of issuing fraudulent certificates will be available, dependent who is responsible for the logs.	Users will still be able to choose which CAs to trust and have control over that. However, dependent on the design of the software of browser vendors, users may not be able to visit domains with a self signed certificate.	Domain holders need to keep monitoring the logs of certificates. Browser vendors or CAs, dependent on the elaboration of the solution, will need to maintain the logs, revocation lists and their updates.
Sustainability	**Availability**	**Educational costs**
Certificate Transparency is dependent on the technique used to create the logs efficiently: Merkle trees. It is important that the technique used for this has second-preimage resistance. Similar dependency is how the revocation lists will be created.	To have this solution fully operational, changes to the browser software of end users and CAs must be made. Also, the faith of self signed certificates is not certain.	Domain holders need to be educated in order to understand how to monitor the logs.

Juridical	Limits	Losses and profits
- The company or organizations responsible for the logs and revocation lists should be held responsible if something goes wrong with the list of log they are responsible for. - Using this solution, governments will have little to none influence on the issued certificates, since domain holders will be able to monitor issued certificates from all around the world. - The question of who will be responsible if a fraudulent certificate is issued, added to the log and not found or indicated by the domain holder: the CA, browser vendor, or the domain holder.	- The solution may not allow self signed certificates anymore. - It will no longer be possible to have valid certificates without the rest of the world being able to check on them.	Dependent on browser vendors, self signed certificates may not be usable anymore, which means that everybody needs to buy a certificate, which will deliver more profits for CAs and a little loss per users.

5.6. DNS-based Authentication of Named Entities (DANE)

DNS-based Authentication of Named Entities (DANE) is another solution for the problems in the current PKI, which is developed by an IETF working group. The goal of DANE is to establish cryptographically secured communications for discovering and authenticating public keys by distributing certificate information through DNSSEC. DANE puts more responsibility at the domain owners, where domain owners can specify limitations about which certificate should be used to connect users to the their website. They can do this by publishing the hash of the certificate they want to be used through DNSSEC. This provides an extra layer of security for clients by ensuring that the certificate sent by the TLS server is indeed the certificate the domain owner wants the client to use. If a web browser, supporting DANE, detects that it is not using the specified certificate, it warns the user, even though perhaps a valid certificate is shown to the user.

To understand DANE, basic understanding of DNSSEC and DNS is needed, which first will be explained here. **DNSSEC** is an extension of the existing DNS, which is the application layer responsible for assigning domain names and mapping these names to IP addresses. To establish an secure communication using the TLS protocol, clients start by exchanging messages with a TLS server, where the server's

IP addressed is looked up using DNS. The client then begins a connection to a particular port at that address and sends the initial messages. At this point however, the client does not know whether an adversary is intercepting and/or altering the communication before it reaches the TLS server or if the initial message was ever received by the TLS server.

Unfortunately, security was not included in the original design of DNS, allowing these kind of simple, yet powerful attacks called DNS spoofing and cache poisoning. An attacker can use these attacks to return false IP addresses to the victim's computer, causing the traffic to go through the attacker's computer. Since the first response of the TLS server may contain a certificate, to allow the authentication of the communication, it is necessary that the client can be sure it is communicating with the right TLS server. The certificate, shown by the TLS server, may even be a valid certificate that is obtained by the attacker through a compromised CA. Users will not notice they are being attacked and proceed as used to when they are for example asked for their login credentials, as happened to the Iranian users when false DigiNotar certificates were being used [Pri11].

After the publication of Steve Bellovin paper in 1995 [Bel95], exposing this problem, the IETF started working on DNSSEC to protect clients from these forged DNS data. After many complications and fundamental modifications to the original DNSSEC, it is now being implemented and is already supported by many software, including the browser Chrome [Lan11d].

In DNSSEC, all DNS answers are signed using public and private key cryptography, which clients can validate by the public keys in the DNSSEC chain until the trusted root is reached. Like the CA system, DNSSEC has a hierarchical trust scheme. However, unlike the many root CAs available in the current PKI, DNSSEC has got only one root that needs to be trusted by all clients. The root is currently managed by ICANN, a nonprofit private organization in the United States. The root is responsible for signing the Top Level Domains (TLDs), such as .com and .nl. The organization managing a TLD is called a registry. The registry for the .nl zone is managed by SIDN. Registries can issue and register domain names for their zone. The registry, in their turn, sign the next layer in the DNSSEC hierarchy, which are the authoritative name servers, such as surfnet.nl

There are two key pairs per zone in DNSSEC, a long term key and a short term key, respectively called the **Key Signing Key (KSK)** and the **Zone Signing Key (ZSK)**. The KSK is used to sign the zone's own ZSK, where ZSK is used to sign the KSK's of zones lower in the hierarchy. Similar to the CA system, only the DNSSEC root can sign its own KSK with its private key. This is shown in Fig. 5.2, where starting at the DNS root zone, the KSK is used to sign the ZSK of the root, which is used to sign the KSK of the TLD zones. The KSK of the TLDs is again used to sign their ZSK, which is then used to sign the KSK of for example surfnet.nl. The KSK of surfnet.nl is then used to sign the ZSK of surfnet.nl, which they can use to sign domains such as example.surfnet.nl. This prevents an untrustworthy signer from

compromising anyone's keys except those in their own subdomains. The actual signing, a similar process that is currently used to sign certificates, is explained shortly in Sec. A.1.3.2.

When using DNSSEC, identifying information about the zone is given to the client, where by trusting the root's KSK, the signed data can be verified from the lowest zone until the root. It was mentioned by the interviewee in Sec. A.1 that in future stages of DNSSEC, this might be built-in automatically into the operating system or browser software of the user. Signing DNS records of .nl domains is already available by the SIDN since 15th of May 2012 in the Netherlands [SID12]. Although not all registrars support this yet, in August 2012, there were more than 600.000 domains signed with DNSSEC in the Netherlands [vM12]. The deployment of DNSSEC will grow even further in the Netherlands, because the government has put the implementation of DNSSEC on the 'pas toe of leg uit'-list since June 2012 [Sta12]. The list contains standards which are obligated by law to be implemented by government organizations. When a standard has been added to the list, government organizations must have a good reason and explain why they are not implementing it.

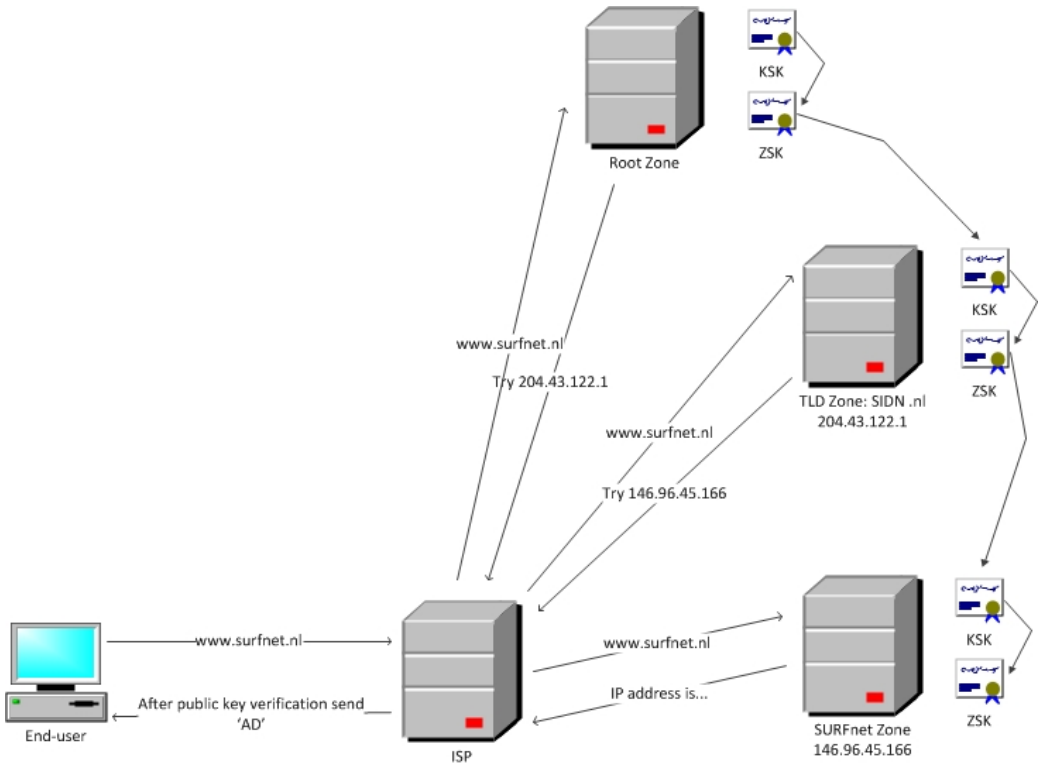

Figure 5.2.: DNSsec scheme

This extension of DNS prevents the previously explained DNS spoofing and cache poisoning attacks, because all answers are signed and can be validated by trusting the DNSSEC root. **DANE** uses DNSSEC to expand security to a further level by adding the extra layer of confirmation to the existing PKI. In the currently used

PKI, the client validates a certificate by trusting CAs.

The single point of failure in the current CA system is that all CAs can issue certificates for domains, without the consent of the domain owner.

In DANE, the same trust chain and technique of DNSSEC is used to provide information about which certificate is supposed to be used for a domain or the domain's services [DNS12].

For DANE, a new record type, called **TLSA**, is used to verify a TLS server certificate or public key with a domain's provided information in the TLSA record. TLSA record consists of four data fields, namely an usage field, a selector field, a matching type field and a field for the certificate association data, see Fig. 5.3.

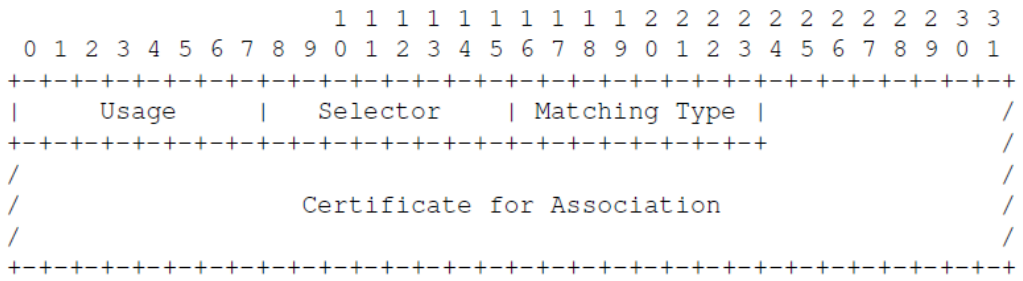

Figure 5.3.: TLSA record data. The *usage*, *selector*, and *matching type* fields are 1 octet. The *certificate for association* has a variable length.

The *usage* field indicates which association will be used to match the presented certificate in a TLS handshake. DANE allows multiple specifications for this: CA constraint, service certificate constraint, trust anchor assertion, and domain-issued certificate [HCS+12]. *CA constraint* means clients should only accept certificates issued by the specified CA. *Service certificate constraint* means clients should only accept a specific certificate from a specific CA. In both, CA constraint and service certificate constraint, the CA issuing the certificate must be trusted by the user. The *trust anchor assertion* can be used when the domain owner has another trust anchor, for example, if the domain issues its own certificates under its own CA and are not expected to be trusted by users. The last usage, *the domain-issued certificate*, allows self signed certificates without involving a third party CA. The self signed certificates are signed by the domain owner's DNS ZSK or a party that will take this out of the hands of domain owners.

To match the presented certificate by the TLS server against the TLSA record, the *selector* field specifies which part should be matched. The choice between the full certificate or the SubjectPublicKeyInfo can be made. When the full certificate is used, the CA can issue a new certificate for itself, without the domain's administrator having to change the TLSA record. Using the SubjectPublicKeyInfo, a domain's administrator can change CAs, keeping the same key, without the need to update the TLSA record.

The *matching field* specifies how the association data should be matched to the certificate from the TLS server. This can be either by public key, SHA-256, or SHA-512.

The *Certificate for Association* contains the bytes to-be matched, provided by the domain owner, is called the TLSA certificate association. The certificate association data can be either the raw data of the full certificate or be its SubjectPublicKeyInfo.

5.6.1. Analysis

The greatest advantage of DANE is that unlike the CA system, domain owners can specify characteristics of their certificates, limiting the chance that a certificate from a trusted compromised CA will be seen as valid by users. In the current PKI, there is no limitation for CAs, where they all have the same privileges when they are trusted. With DANE's limitation of trust, the solution is able to prevent incidents similar to DigiNotar, where a compromised trusted CA issues fraudulent certificates for another domain in order to eavesdrop users.

The solution does require the deployment of DNSSEC, which has already begun and as the facts about its progress shows, will probably only increase. To deploy DANE in DNSSEC, only a small change in the DNSSEC's server software is needed and will cost only a little after the deployment of DNSSEC has been completed. The most important cost, after DNSSEC has been deployed already, will be for server administrators for understanding DANE and DNSSEC, where DNSSEC can be quite complex for creating and maintaining its KSK and ZSK. Domain owners will also need to understand how they can add restrictions to the usage of certificates in their domain and maintain this restriction.

Also, having DNSSEC as the basis of DANE, it is important to understand that DANE's security relies on the security of DNSSEC, which is still under discussion. Because DNSSEC's difficult configuration with its KSK and ZSK, mistakes are easily made. This results into DNS failure and the unavailability of a domain. Comcast, one of the biggest home Internet service providers in the United States that supports DNSSEC, has created a website showing domains that fail DNSSEC validation [com]. Domains can fail to validate their DNSSEC for reasons such as signature expirations, KSK rollover failure, and inconsistent ZSK.

The great advantage of DANE has, in my opinion, overdone it by creating the ability to trust self signed certificates of domain owners. Although this can be seen as an advantage of DANE, where domain owners can create trust with their users, without the need to buy an certificate from a CA, it is also the creation of this solution's new single point of failure. The allowance of self signed certificates, which is signed by the domain owner's ZSK and placed in their DNSSEC server, will replace the decentralized trust in the current CA system towards a centralized trust, managed by ICANN. For simplicity, lets call DANE with the single point of failure, **DANES**, which is DANE with the additional allowance of self signed certificates.

If DANES continuous to be deployed, the complete Internet will be dependent on an organization in the United States, where their government can in theory even take over the control. As the interviewee in Sec. A.2 mentioned, governments would not want to be dependent or possibly controlled by another government.

Furthermore, allowing the deployment of DANES will put ISPs and domain owners as the new target for attackers. It is only logical to assume that the security in a Certificate Authority is better organized and kept up to date than most domain owners will be able to do. Another point of attention would be if CA organization and TLDs would merge, specially their networks, allowing a compromisation to issue certificates and DNSSEC keys.

Another security risk of DANES is that it will become very easy for attackers to setup a secure DNS server for some domain, similar to a valid domain. An example of this would be *paypa*1.*com*, were the last letter is replaced by a number. The by attackers created website, will be able to look exactly the same and be accepted by users, because of the domain's valid DANE record.

Therefore, having DANE as an extra layer of security, is indeed a possible solution to improve PKI, but only if both their usages is obligated and they will work independent of each other. This would result into two different kind of authorities, DNSSEC root and CA root, that guarantee the validity of a public key. But as we have seen, DANE's powerful support layer has created DANES, which wants to replace the current CA system and handle trust independently.

Hoping that the DANE workgroup will remove the aspects that result in DANES, we can move on to other considerations of deploying DANE. The first involves the additional latency, since two verifications are needed, the CA system and DNSSEC, before a domain is verified.

5.6.2. Solution aspects

This is a summary of the various aspects of DANE, grouped by type, mentioned in the previous paragraph of the solution analyst, according to the aspects discussed in Sec. 1.3.1.

Security	Usability	Costs
Risks	**Efficiency**	**Transition costs**
- Centralized trust: ICANN will get the power over Internet and will be able to shutdown any country as it desires, since Internet is dependent on DNS. - Creation of DANES: single point of failure, making ISPs and domain owners the new target for attackers.	The system will experience some latency because of DNSSEC usage, obligating two different verification sources for clients before they can access a domain.	- DNSSEC servers and the expensive HSMs to keep the KSK and ZSK keys secure. - DANE itself requires little changes after DNSSEC is deployed. DNSSEC deployement has already started, but is known to cost a lot due to the many changes. DNSSEC brings along changes in all possible levels, since DNS is basis of Internet.
Improvements	**Control**	**Maintenance costs**
- Problem of DNS spoofing is solved - PKI is divided into domains, limiting the risks to only that domain when a CA is compromised or becomes fraudulent.	Similar to the current system, users can revoke their trust in CAs, but will loose access to some parts of Internet. With DANE, users are obligated to trust the DNSSEC root, ICANN.	- DNSSEC servers will need much maintenance due to key change. - Domain owners must keep an eye on their TLSA records.
Sustainability	**Availability**	**Educational costs**
DANE is dependent on DNSSEC.	The previously available resources will still be used since the CA system will stay intact.	Administrators of DNSSEC, domain operators, need to be educated in order to be able to perform the advanced key management that is requires in DNSSEC and maintain their TLSA records.
Juridical	**Limits**	**Losses and profits**
- ICANN is the main and only root in DNSSEC, giving them great power over the Internet. - Domain owners will be responsible for the maintenance of their own KSK, ZSK, and TLSA record.	DANE can only be used for domains, which have a DNS record.	The losses for deploying DANE is the same as the losses and costs for deploying DNSSEC. However, if DANES is deployed, CAs will suffer huge losses, not to mention the creation of the single point of failure.

5.7. Multiple Certificate Signatures (MCS)

Here, I will mention another idea to improve the trust in PKI, which was given to me by a college at Deloitte. In a short discussion of how to improve PKI, he mentioned signing a public key by three CAs instead of one CA as a solution. This means that PKI would stay the same, but would gain increased security. As I have not been able to find any literature containing a similar idea, I want to mention this idea's advantages and disadvantages in order to help others with a similar solution in mind. The solution has received the straight forward name: Multiple Certificate Signatures (MCS).

The first advantage of MCS is that even if a Certificate Authority would become compromised, the trust in the public key would still hold, since there will be at least another trusted CA that has signed the same public key. Public keys, signed by only one CA, should eventually not be trusted by users. In the DigiNotar or Comodo's case for example, the fraudulently created certificates, which were only signed by one CA, would not have been trusted by users. DigiNotar would then still be removed from the trusted CAs, but at least the user's privacy and data would have been protected.

MCS would show its second great benefit if the public key's were signed by at least three CAs. In that case, the owner of the public key will have enough time to get the certificate signed by another CA, keeping the trust in the public key, since it is still signed by two other CAs. The solution can even remove an organization or country's power for issuing fraudulent certificates, by only allowing certificates to be signed by three different CAs, from different root CAs, all from different countries.

The second property, where the CAs are all from different root CAs, must be made mandatory for this solution. By disallowing the three different CAs being from the same root CA, prevents root CAs becoming the target of attacks. Attacking the root CA would however not allow the direct creation of a valid certificate, but would remove the second benefit of this solution, because all CAs would be distrusted when the trust in their root CA is revoked. Therefore, the three CAs must all be from different root CAs. Creating a compromised certificate by a hacker, organization, or government, would become very difficult. To create a fraudulent certificate, the attackers are then required to compromise three different CAs, from three different countries.

To allow the true independence of the three different CAs, it must be verified that they are all from different root CAs and do not have overlapping CAs in their certification paths. If the three CAs would have some other CA, higher in the hierarchy in common, compromising that CA would have the same effect on certificates from MCS as it has on the current certificates when one CA is compromised. The easy solution to solve this would be by disallowing cross certification by CAs, resulting in a straight path from CAs to their root CAs. This can be a frightening solution for CAs, which have strengthen their position by having their public key signed by

multiple CAs. Further research will be required in order to confirm whether removing cross certification is the best solution. Either way, for this solution to work, it is important to have no overlap between the CAs, or the method will not be an improvement.

The solution has some disadvantages, with the most important one being the many changes that will be required and the costs it would bring along. Proving and showing to users that a public key is signed by multiple CAs, can be made possible through three different ways. The first method would be by having multiple certificates sent to the user's request, where the second method would be by having one certificate that would contain multiple signatures for the certificate's public key. The third possibility would be to sign a public key by multiple CAs using a shared key, where each CA would have its own private part of the shared key.

Since X.509 assumes only one issuer per certificate and does not support cross-signing of public keys [PE11], the first and second approach would require changing the browser softwares, how certificates are processed in the many third party softwares, and the structure of the certificate standard X.509. If the certificate structure can be changed such that multiple signatures are allowed in one certificate, and can be changed independently when one of the CAs are compromised, then the second method would be preferred due to latency issues when comparing to the first method.

The third approach, containing the cryptographic secret sharing solution, would not require any change to the current softwares and X.509, but would require great communication skills between CAs in order to sign a public key with a shared key. Since clients requesting a certificate, will still want to choose their CAs themselves, every possible combination of three CAs will need to have a shared key with each other and be trusted in the browser softwares and Operating System. The trusted list of public keys in browsers and Operating systems will become huge, since every combination of the shared key must be available. The possible combinations for certificates signed by multiple CAs, starting at three CAs, would be $\sum_{n=3} \binom{c}{n}$ where c is the number of trusted CAs. The amount of combinations for three CAs would already be 35820200 possible keys, which is 59700 times more than the currently 600 trusted CAs public keys that are saved in the user's browser [Ben11]. This makes the third approach for signing the public key very unrealistic.

Another problem with this solution is the revocation of certificates. Certificates can become invalid when for example the private key is stolen or lost, which is a possible scenario since not all organization keep their private keys in an optimal safe environment. In the perfect world of certificates, users would be made aware of a revoked certificate immediately. This is unfortunately not possible yet, resulting in delayed information about newly revoked certificates. This delay will become even more using this solution, because whenever anybody realizes the certificate has been compromised, multiple organization need to be made aware, including at least three different Certificate Authorities.

A point to mention is the increased cost for certificate owners, since they will need to pay multiple CAs to obtain a valid certificate. However, it is possible that the cost for a certificate will decrease since CAs clients will increase due to this solution's requirement of having at least three CAs signatures.

The advantage about this solution is that the single point of failure, which currently lies at the Certificate Authority issuing the certificate, is removed. However, to keep this intact, it is important to disallow companies that are willing to perform as a middleman for taking care of the certificate's signatures by multiple Certificate Authorities for other companies. This would, as the interviewee mentioned in Sec. A.4, take away the benefits of having MCS, because the middleman will become the new single point of failure.

5.7.1. Solution aspects

This is a summary of the various aspects of MCS, grouped by type, mentioned in the previous paragraph, according to the aspects discussed in Sec. 1.3.1.

Security	**Usability**	**Costs**
Risks	**Efficiency**	**Transition costs**
- Possible revocation latency due to informing multiple CAs. - Middleman must be prevented in order to disallow the new creation of a single point of failure.	- The received certificate size will at most be three times larger, when signed by three CAs, which may add some latency, slowing down the connection. - Companies will need to request for multiple certificates, which will make administrative work more difficult and take more time.	This depends very much on which approach is taken to allow certificates to be signed by multiple Certificate Authorities. Probably browser software, third party software, and the X.509 structure needs to be changed, since the third approach is not very realistic. The cost for this will therefore be probably more than any other solution discussed so far.
Improvements	**Control**	**Maintenance costs**
- One compromised CAs will not put user's privacy and data in danger anymore. - Certificate owners will keep a valid certificate, even when on of the CAs has been compromised, and will have enough time to replace it with another CA's signature. - Governments will no longer be able to misuse their powers and issue fraudulent certificates.	The control of this solution will not be different from the current method, where users are given the choice to distrust a CA. Since there will be three CAs signing a public key, the choice of distrusting a CA will not immediately results into inaccessibility of domains.	The same as the current PKI, however, storage will increase by three times, which can bring along some costs for large companies.

Sustainability	**Availability**	**Educational costs**
The solution has the same characteristics as the current solution, which will stay operational with DNSSEC or new protocols replacing for example SSL.	Additional harddisks will probably be needed since everything will take three times more space. Everything else will stay available and can be reused.	Depends very much on which approach is take to allow certificates to be signed by multiple Certificate Authorities.
Juridical	**Limits**	**Losses and profits**
The power of government and large CA companies is restricted and more spread due to the multiple CA signatures that are needed, each from another country.	There will be no additional limitations in comparison to the current PKI.	- CAs will probably earn more money since they get more customers - Certificate holders will need to pay more for a valid certificate.

6. Conclusion

Game security

Concerning the security in games, the research and interviews have shown that as long as the player possesses the game software, modifications to the game software make cheating possible. As we have seen in Sec. 4.2, there are many solutions to make cheating more difficult. For most game companies, especially small ones, this costs too much time and effort to deploy. The research shows that for these companies, it is all about finding the balance between the amount of cheats and the costs to reduce it. This puts game hackers in an advantage position, especially for small game studios. Neverthless, security measurements only lower the amount of cheaters, but does not fully remove them. Statistically monitoring seems to be the only solution to detect cheaters. Unfortunately, many game studios can not afford to implement such features.

Internet Public Key Infrastructure

The most important problem of the current PKI is that all CAs that are trusted by browsers or operating systems, have the same privileges and can all issue certificates for any domain they want, without the need to make their issued certificates public. As most users do not change their preferences, they are almost obligated to trust the CAs to not abuse this ability or become compromised, which can be used to eavesdrop users in combination with a MITM attack. As in the current PKI, the issued certificates by a CA are not transparent and controllable in real time, it is not possible to detect abuses of the privileges, until it is noticed by a user that has enough knowledge to understand he or she is being mislead. DigiNotar is only one of the cases that was exposed and removed from the list of trusted CAs. Who knows how many other CAs, who may not even be aware of it themselves, might have been compromised.

Since the DigiNotar incident, the world became aware that the theoretical attack on CAs could actually happen. This motivated many to think of a solution to improve PKI and remove this risk where all CAs have the same privileges. As discussed in chapter 5, all solutions that were explained and discussed dot not come without problems. Each solutions introduces new security issues, usability problems, extra costs, or even new single point of failures. As most of the problems and security issues in these designs will be solved in time, a single point of failure will not easily be solved, as we are encountering the same problem with the current PKI. Deploying

a new solution that limits or removes the old single point of failure, but introduces another, will only cost us a lot of money, time, and effort.

As the current PKI's weakness has been shown through the DigiNotar incident, it is not an option to keep the current PKI as it is. Apart of the CA that goes banckrupt, keeping the current PKI as it is, will probably be the cheapest solution. This is, however, not acceptable when we know that the lives of people can be at stake if they are eavesdropped by their government for example. There are more than 600 trusted CAs that can get compromised, if this is not already the case. The Comodo hacker mentioned that he had still access to a couple of other CAs that he could yet abuse, without the CAs being aware of their compromisation [Com11b]. Therefore it is definitely important to come up with a solution to improve the trust in PKI.

After evaluating each of the proposals based on the aspects, which were defined in Sec. 1.3.1, the results show that most of the solutions are not fit to be deployed as the overall solution for the Internet. Let us start by summarizing why Public Key Pinning and Perspectives & Convergence can not satisfy everybody, and therefore, should not be deployed as the global solution.

Public Key Pinning can not be applied to all domains, since it would give browser vendors, managing the pins, too much power over the global trust, where the trust would be centralized. Although it is already the case that browser vendors can determine which CAs to trust, giving them the power to determine which certificates should be trusted, gives them too much power and/or places them as potential targets for attacks or manipulations by the browser vendor's government. Another main problem with Public Key Pinning is its bootstrapping problem, where the user will not be sure about the pins of a domain when it is visited for the very first time by the user.

Perspectives & Convergence gives the more advanced users the ability to release themselves from the CA system and the determined trust anchors by browsers. The disadvantage of this solution is that it can not be used by large companies and governments, because creating a large scale attack, through simply running many notary servers, will be possible.

The *Sovereign Keys* solution, which has been the inspiration for the Certificate Transparency solution, seems to be too complex to deploy and still has many obstacles to bypass. The timeline servers create a new target point for attackers, and it is not yet clear who will manage them. The Sovereign Keys show similarities to Convergence, since the mirrors can be run by anyone, making the solution undesirable for large organizations and governments. Also, the idea of Sovereign Keys is already improved by Certificate Transparency, which has more benefits, its easier to deploy, and has less major risks.

Certificate Transparency supports the current PKI by making it transparent. Certificates issued by CAs will no longer be a secret, allowing domain owners to keep an eye on which certificates are issued for their domains and revoke the ones that are not requested by them. By introducing this control factor, CAs are no longer

able to abuse their power or issue certificates unnoticed when they are compromised. Whether the solution will succeed depends very much by whom the audit logs will be managed eventually. Like Perspectives & Convergence and Sovereign Keys, Certificate Transparency will not be able to become the world wide solution for everyone if the audit logs can be managed by anyone. Most parties prefer to know who they are relying on and who they can held responsible if an incident may occur.

In *DANE*, domain holders are able to add restrictions to which certificates are allowed for their domain, through the trust anchor of DNSSEC. This limits the privilege of issuing certificates for any domain by a compromised CA and allows the detection of abuse by attentive domain holders. A great disadvantage of DANE is if DANE will be used as a standalone solution instead of a support solution for the CA system. In this case, DANE will create a new and weaker single point of failure, due to its centralized trust. The new solution should not create a new single point of failure, since this is exactly what we are trying to repair. Another disadvantage is that domain owners will become the new target for attackers, where manipulating their DNSSEC sever will put their domain out of service. These important disadvantages would be solved if DANE would stay as a support for the current PKI and only allow valid certificates in combination with a valid TLSA record. Having DANE as a support solution, attackers will need to both successfully compromise a CA and domain owners.

The *MCS* solution does not necessarily remove the problems, but makes it more difficult for the hackers and CAs to issue certificates without the knowledge of at least two other CAs. The solution requires many changes to the current system, including software changes of small companies, and will not benefit the companies that use certificates in their costs. MCS does not add any solution for faster detection of fraudulent certificates, but increases the bar. The solution needs to first evolve to the next phase before any conclusions can be made.

Finally, to give a very short conclusion: Certificate Transparency and DANE have, according to this research, the best chances of improving the current PKI. Certificate Transparency allows faster detection of compromised certificates and CAs and DANE allows the limitation of the impact by compromised CAs.

Both solutions are not yet ready for deployement. Certificate Transparency has an unfinished design, with the most important part being the unfinished specification about who will be responsible for the audit logs. DANE is still not deployable, because DNSSEC is not yet deployed. Although DANE does give DNSSEC another reason for deployement, it is not yet clear when this will be the case.

It is advisable that in future research, we first look for solutions to further improve DANE and Certificate Transparency. There are still some major measurements needed to strengthen and improve these solutions before they can be deployed. Another suggestion for further research would be to create a detection system. Both DANE and Certificate Tranparency require domain owners to monitor their domains and take actions in case of compromisation. A convenient research would be to

develop an automatic detection or warning system for domain owners.

The described aspects for improving PKI in Sec. 1.3.1, can be used as a starting point for new proposals. It can help evaluating the aspects and may even help to improve the proposal. However, if the aspects are used in a further stadium of a proposal, the aspects should be made more specific. An example would be to address specifically the amount of latency that is acceptable.

A. Appendix: Interviews

A.1. Roland van Rijswijk and Gijs van den Broek - SURFnet

6 August 2012 at 10:00 - 11:30 in Utrecht

The main reason for this interview was to get more familiar with DNSSEC and the weaknesses of PKI.

A.1.1. Introduction

Gijs van den Broek said he just graduated at SURFnet and was now working at SURFnet in line of his research involving DNSsec. He also works at ZorgTTP, as a colleague of Gerard Tel, where he deals with security issues. Last year he was involved with the issues caused by DigiNotar at ZorgTTP.

Roland van Rijswijk is Technical product manager at SURFnet where he deals with internet security involving many cryptographic aspects.

A.1.1.1. SURFnet

In order to clarify the relation of SURFnet and DNSsec, Roland explained that SURFnet is a NREN, a national research and educational network, meaning they supply internet and applications to educational organizations in the Netherlands. SURFnet was founded in 1986 as a project and became a company in 1988. SURFnet is a non-profit organization who is compensated for their internet service. They also receive subsidies for research and innovation.

A.1.1.2. SURFnet and security

To my question of how important security is for SURFnet, Roland answered that security is very important, because they have a good reputation and have to keep this high. They have (academic) hospitals as clients that are very much dependent on SURFnet. Security is a basic condition they have to take into account. A security compromise could result in serious damage to them. Therefore their network is monitored 24/7.

A.1.1.3. Relation between SURFnet and PKI

The connected institutions to SURFnet all have servers which they need certificates for. SURFnet orders many certificates for their clients and has for this, together with TERENA, a not-for-profit association of European NRENs, called for tenders. The tender was won by Comodo, a big certificate service from the United States and they deliver certificates to SURFnet for a flat fee, enabling SURFnet to order as many certificates as they want.

A.1.1.4. Comodo as CA for SURFnet

I acted surprised when hearing that Comodo was their CA, because Comodo issued fraudulent certificates in 2011 due to a hack. Roland said that SURFnet was not happy about this, but was stuck to Comodo through contract. Fortunately the hack did not affect Comodo as bad as it affected DigiNotar. Comodo is much bigger and has more power in order to prevent the trust in them by CA's higher in the hierarchy. Also, the attack used for Comodo was different from the attack to DigiNotar. Comodo works with a reseller model, and there is only one reseller needed to not function properly to make an attack possible, which was the case in 2011 [Com11a]. A reseller can issue certificates themselves, or dependent on their given certificate, even sign another CA to do the same. In 2011, one of the resellers of Comodo was attacked and their password was obtained by the attackers from Iran, making it possible for them to issue certificates for several different domains signed by Comodo. Fortunate for Comodo, they could improve their processes, which led to the tightening of the identification process for requesting certificates. SURFnet also experienced the consequences of this. The request for a certificate was initially granted within a few hours, however, after the attack it could take multiple days or weeks.

A.1.1.5. Did SURFnet get any problems when DigiNotar was compromised?

Roland explained the problems caused by DigiNotar were limited for SURFnet. However, the much bigger corporation called SURF, where SURFnet is a part of, did have some troubles. SURF manages the Studielink website using DigiNotar certificates. They, Roland said, had to deal with this situation, but SURFnet was fortunately not affected by DigiNotar.

A.1.1.6. What would the problems be if Comodo's certificates were revoked?

Roland said that fortunately, Comodo knew which reseller was compromised and their certificates were revoked immediately. After an audit they confirmed other certificates were not compromised and could stay valid. If all certificates were revoked, it would be disastrous. Gijs explained that certificates are not only used in websites, but also in programs. In the tax program for the government for example

a certificate is processed and if all the certificates from DigiNotar were revoked immediately, all these programs would not work anymore. It is very complex and time consuming to change these certificates.

A.1.1.7. Does SURFnet has a backup plan?

Roland said that whenever Comodo is compromised, SURFnet will change their CA as soon as possible. It will of course take a while and they will give priority to the more important organizations, such as hospitals or certificates for the government. However, the person knowing more about this subject is on not available right now.

A.1.2. Current PKI model

Roland said the weaknesses of PKI can be divided into two parts: technically and behaviorally. In the technical part, he thinks that PKI is not even that bad. The technical part for the secure connections with TLS and SSL has some minor issues, but in general it is patched very well. However, if you look at the behavioral part you will find many problems. There are couple of hundreds authorities who are able to issue certificates, and all of them are handled by the browser and there is only one CA needed to make mistakes. If all the CA's would keep their end save, Roland said, the system would work fine.

A.1.2.1. The beginning of PKI

When this model was designed, internet was very small and the designers didn't take into account that it would become so large. In the time the model was designed, the amount of certificates in use was very little. Besides, everyone knew each other and trust was not an issue. When e-commerce grew, however, certificates became big business because all the websites using payments needed a certificate. At this point competition occurred by pricing the certificates. However, Roland feels that the competition in pricing went at the expense of the audit process where CA's needed to check if a certificate owner was the rightful owner for the certificate. Sometimes the CA doesn't do any checks at all to save money. This led to the classification of certificates depending on the level of trust. The problem, however, is that the end-user is not able to keep track of these classifications. The only information given to the user, by the banks and governments, is that there needs to be a padlock icon in the browser. However, with this padlock icon the user can not see what behavior process was used to get this certificate and what policy the CA issuing the certificate uses. This is where the system has gone wrong and this is almost not reparable anymore. Clients of CA's will not accept it if CA's would raise the prices of certificates.

Roland said that there are now too many CA's and gave the Chinese government, a trusted CA, as an example. Something to think about he said, because how reliable

is the Chinese government? And what are their interests. All these issues remove the correctness of the model.

A.1.2.2. Weakness of PKI

After his explanation I said that I think it is a mistake in the technical part of the model that if one CA is compromised, the complete system is in danger. Gijs answered that if all of the CA's meet the requirements, then the model is great. However, everyone is indeed dependent on the weakest link. The only solution in the model is to remove the compromised CA. This is also what they wanted to do with DigiNotar, Roland said, however, the Dutch government complained as long as needed in order to delay the removal of DigiNotar. They said they foresee great problems for the infrastructure in the Netherlands if DigiNotar was not trusted anymore. This is an example of behavior, Roland said.

A.1.2.3. Testing model for a CA

Two interesting properties of the current PKI model were pointed out by Gijs with the first being that every root CA can issue an certificate for any domain, anywhere in the world.

The second property he mentioned, is that there are only two requirements needed to become a CA. The webtrust certification model through audits is the only requirement needed to show the padlock icon to the user properly. The audits for this can only be done by a small amount of companies who are paid by the CA to do the audit, therefore the reliability of the report can be questioned. Recently, the Extended Validation (EV) model was added, which requires more verification and requirements for a CA and the entity requesting the certificate. However, the EV is much more expensive and the user does not see or know the difference between the padlock icon and the extra green bar in the browser.

A.1.3. DNSsec

To clarify how DNSsec works for myself, I asked if they could explain who actually signs the DNS records using DNSsec.

Roland van Rijswijk first began to explain how DNSsec works while Gijs was drawing the DNSsec scheme, very similar to Figure Fig. 5.2 on the white board. Roland explained that the trust scheme of DNSsec is, just like the PKI, hierarchical and starts with a DNS root zone. In PKI there are multiple root CA's which can be trusted by the end-user or the browser organization, however, in DNSsec there is only one root zone, which needs to be trusted by end-users and is managed by ICANN. The KSK of the root zone can be compared to a root CA certificate.

After the DNS root zone comes the authoritative name servers for top-level domains in the hierarchy. A top-level domain (TLD) is granted by ICANN and is the last part of an internet domain name such as .nl, .com, or .org. The organization managing a top-level domain is called a registry. The registry for the .nl zone for example is managed by SIDN. The registry can issue and register domain names for their zone.

After the TLD's comes the authoritative name servers in the hierarchy, such as the one for surfnet.nl.

A.1.3.1. Key Signing Key and Zone Signing Key

The public key cryptography of DNSSEC works with two keys, a long term key and a short term key, respectively called the Key Signing Key (KSK) and the Zone Signing Key (ZSK). The KSK of a zone must be signed by the ZSK of a zone higher in the hierarchy. Only the DNS root can sign it's own KSK with their private key. The ZSK is used to sign the entire zone lower in the hierarchy, where the KSK is only used to sign it's own ZSK.

Beginning at the DNS root zone, and as is showed in Figure Fig. 5.2, the KSK is used to sign the ZSK of the root, which is used to sign the KSK of the TLD zones.

The KSK of the TLD's is again used to sign their short term ZSK. The ZSK of a top-level domain, such as the SIDN for .nl domains, is then used to sign the KSK of for example surfnet.nl, which is one step lower in the DNSsec hierarchy. The KSK of surfnet.nl is then used to sign the ZSK of surfnet.nl.

A.1.3.2. Signing a domain

After the explanation about the KSK and ZSK in DNSsec, Roland talked about who actually signs the DNS records of a domain. He explained that registries are the organizations behind a top-level domain such as SIDN for .nl, and that registrars are organizations that communicate with the registries to apply for a domain for a registrant, the domain owner. The public and private key, which is used to sign the DNS record, is created at the server for that domain. The server can be at the internet hosting server or in possession of the domain owner. The created public key is then given to the registrar whom in turn will communicate with the registry of that zone, e.g. SIDN for .nl, in order to give the registry the public key for that domain. The SIDN will sign the public key with their ZSK after acceptance.

A.1.3.3. Working of DNSsec

Roland and Gijs then explained about the working of DNSsec. They explained that whenever an end-user needs to know the IP address of a domain name, the end-user will request this through the cache resolver of the associated Internet Service

Provider (ISP). The domain name surfnet.nl was used as an example in their explanation. For DNSsec to work, it is necessary that the right KSK of the DNS root is trusted by the resolver. The resolver will first communicate with the DNS root, asking for the IP address of surfnet.nl. If the root does not have the answer, it will give a signed answer with the nameserver address of the top-level domain .nl, the zone where surfnet.nl is in. The ISP will then ask the TLD nameserver, in this case the SIDN, for the IP address of surfnet.nl and if SIDN does not know the IP address, it will return a signed answer of the authoritative nameserver address of surfnet.nl. The ISP will then ask the nameserver of surfnet.nl for the IP address. The nameserver of surfnet.nl then returns a signed answer of the IP address.

Validating the IP address can occur using the public key of the DNS root. The resolver can use this public key to validate the TLD's signed public key. In the same manner, the verified public key of the TLD is then used to verify the signed public key in the answer sent by the nameserver of surfnet.nl. After these verifications, the ISP is certain it has the right answer to the end-user's request for the domain name surfnet.nl. The IP address is then returned to the end-user containing the letters 'AD', meaning the verification was OK.

Roland explained that the end-user and the cache server only need to know the KSK of the DNS root. It is possible that this might be built-in automatically by the operating system of the user in future stages of DNSsec.

After the explanation Gijs said that the signing of domains already is begun and that it is rapidly growing for .nl domains. Roland said he was curious about how the changing of the Key Signing Key of the DNS root will be, which must happen in the near future. It will probably result in the dysfunction of some systems.

A.1.3.4. Weakness of DNSsec

Roland said that the exchange and acceptance of keys between the DNS root and TLD's is between a small group of people who know each other through yearly meetings in ICANN. The protocol for controlling for this group is organized properly and is witnessed by the Trusted Community Representatives. The weakest link, Roland said, is between the top-level domain organizations and the registrars.

The exchange and acceptance of keys made by registrants and given to the registrar is, however, vulnerable because there is competition between registrars. Therefore a registrar must deliver a domainname for a low price. Performing an integrity check for the DNSKEY records will make the process more expensive. In theory for example, the registrars are supposed to at least double check the public key of the registrants, but in practice they will probably copy-paste the public keys. Roland also said that due to the complexity of DNSsec, people might make mistakes more easily.

A.1.3.5. Spreading the risk

A year ago Roland was present at Defcon 2011 in Las Vegas where he attended to the presentation of Moxie Marlinspike, the pseudonym of the computer security researcher, against man-in-the-middle attack. The idea is to have multiple servers across different regions in the world and when an end-user wanted to communicate with some domain, the end-user could check if all of the servers return the same certificate. If one of the servers returns different value for example, then the man-in-the-middle attack is used on that server. This way the risk the end-user has, is spread.

Using this theoretical model omits the use of CA's, because now every server could sign it's own DNS or domain and the trustfulness of it could be confirmed using the different servers. Roland said he likes to use the idea as an addition to the current protocol, but that he wouldn't choose to omit the CA's. It is better to spread the risk he said.

A.2. Program Manager Information Security - Governmental organization

27 August 2012 at 10:00 - 11:30

The purpose of this interview was to get the interviewee's opinion about the weaknesses of PKI and possible idea's for improvement.

The interviewee A. is the Program Manager Information Security at an government department. A. is also the CSP manager of PKI, which means that he is resposible for all of the services in the department around PKI. He manages the people who execute tasks around PKI and ensures that there is an infrastructur in order to use PKI.

At the department, the interviewee said, certificates are used for websites, the link between networks and personal passes. They are also going to issue certificates for machines.

PKI versus PKI-Overheid

A. said that the technique used in both is the same, and PKI-Overheid has the same problems as any other PKI systems when a CA is compromised. The only difference is that in PKI-Overheid the government is responsible for the highest root CA. This was agreed by the cabinet in 1999.

DigiNotar

When DigiNotar was compromised and it turned out the trust in PKI-Overheid was also in danger, Microsoft and Mozilla were asked to keep the trust in PKI-Overheid,

otherwise the Netherlands would not be able to communicatie digitally anymore, he said. The impact for the department was very small, he said, because they only had a small amount of certificates issued by DigiNotar which could easily be replaced.

Audit reports are first shown to the CA

The interviewee said that if the audit reports are made public, it will be magnified by the press. He also said that more audit will not necessarily solve the problem.

Weaknesses and problems with PKI

A. said that it is unfortunate that with the hack on DigiNotar, due to measurements, PKI became unnecessary more expensive and brought a lot of expenses along for many companies.

He also mentioned the problem of responsibility in the current PKI where commercial companies such as Microsoft and Mozilla are in a position where they can be held responsible, no matter what they do. A. said that if these companies choose to keep the trust, companies can claim compensation for damages, and if they distrust a CA, they still can be held responsible for their actions.

He believes that the government should not be dependent on a commercial company as is now the case with Microsoft, Mozilla and other IT suppliers. Even with DNSsec, he feels ICANN should not be able to turn off the complete internet of the Netherlands, which will be the case with DNSsec. However, if the country's government would be in charge of the country's internet, then the danger exists that a government would abuse this power. Therefore there should be some international agreements about this, he said.

Another point is that the keylength, now 2048 bits, is getting bigger, which requires a lot of computing power. However, migrating to elliptical curve is still too expensive due to the license.

The EV-model

The interviewee was not impressed by the EV model and said that it is nothing more than another look at the requester.

Future of PKI at the governmental organization

At the end of the interview, he showed me an interesting presentation about a possible new PKI infrastructure where the main CA, signed by the PKI-Overheid CA, is held offline and signs the online CA of the department. The new infrastructure will be used to communicate with third parties and to allow trusted third parties inside their network. In this infrastructure, access to users will be given on a role based method.

Moxie Marlinspike

A. said that such system, where the trust is dependent on unknown parties, is not desirable for governments. If the nodes in between are formally organized, it could work. However, if, during an important transaction, many negative votes are

received, the trust will be gone which is not acceptable. He said that such method probably will be good for citizens and probably for 90% the of businesses, but for 10% or so, it will not be enough.

Blacklisting and whitelisting

With a whitelisting next to a blacklisting, we can know which certificates are valid, he said. According to the interviewee, having a whitelisting could have solved the problem with DigiNotar, because then they could have known which certificates were issued and then distrust only those certificates and not the complete CA.

Sign certificates by three different CA's, from different roots

The interviewee's first reaction was that this is a complicated solution, but an interesting idea. A. said that for every security step, you create a new attack factor and that I should look for this attack factor.

A.3. Cooperative Information Security Officer - The Company

31 August 2012 at 11:00 - 11:30

The main goal of this interview was to get the interviewer's opinion about PKI and my possible solutions to decrease the risks in PKI.

The interviewee B. is a Cooperative Information Security Officer at a large company in the Netherlands. B. currently does not work directly with PKI, but has worked on infrastructures using PKI in the past. At their company, PKI is mostly used to secure communication between servers and machines. The machines also make use of PKI. And of course, communication between servers and users, mostly websites, is also secured through HTTPS/SSL.

Problems with PKI at the company

The main problems are in the certificate management process. At the time DigiNotar was compromised, the company had no overview of their certificates and CA's, resulting in a period where they had to find out if they actually made use of DigiNotar certificates. Fortunately, they did not have any certificates issued by DigiNotar. Another example is the validity period of certificates which causes trouble when the certificate is not renewed in time. Also the change in key length by Microsoft, not accepting short keys, had some consequences.

Although there is still no backup CA or plan at the company, the DigiNotar incident did make people more aware of the problem and the need for an overview of their certificates and a solution for when another incident happens.

Weakness of PKI

B. was not sure if it is a weakness of PKI in the case one CA is compromised, all domains are in danger. The essential point of PKI is, he said, that the trust chain must have an anchor somewhere and whenever this point of trust fails, the complete infrastructure will fail.

According to him, PKI is basicly the method that shifts the problems to another level when it starts getting difficult, beginning at the exchange of messages, encryption using public key, and ending with issuing certificates by CA's.

One root CA (DNSsec) or multiple root CA's

According to the interviewee, the choice for one root or multiple roots, is dependent on the amount of information one wants to protect. For the government one central CA is preferred. This is not because of the trust in one CA, but the trust in the organization. However, for commercial organizations, one CA causes organizational problems: who will be responsible, able to manage or will pay for the CA. In PKI, the responsibility is the biggest aspect. The CA's issue the certificates, but in most cases you can not claim compensations there.

B. also gave the example that gas, water or electricity are delivered by multiple different companies in the Netherlands which are audited by the government. Also, in the context of availability, if anything should happen to one company, with multiple CA's, we can always switch to another one.

Improvements for PKI

The interviewee mentioned that users are currently not enough aware of how PKI works. An improvement would be if PKI worked without any knowledge or interference of the user. Just like electricity, if it works, it works and otherwise the user notices that it doesn't work. However, if this is the case with internet, like in DNSsec, it will bring other problems along.

Attention points implementing new model

B. said one of the aspects is transition cost where you need to know how much effort and costs will be involved. Another aspect is the operational costs where you need to know the costs, the level of user friendly and the difference between the security levels. He also mentioned that with every new model, one must think how to guarantee the integrity and authenticity over the information.

A.3.1. Possible solutions to PKI

3 CA's signing the certificate instead of one CA: This will give more robustness and improves the switch period, but there are some costs attached to it. You will have to look at what kind of costs these are and in which cases this solution will be applied.

Use of white listing next to blacklisting: B. said that both are very much the same, if a certificate is on one of the lists, you know it will not be on the other list.

The only difference is the size of the list, where publishing the smallest would better. He also mentioned using OCSP for the lists, which only give answers to requests whether the certificate is valid or not.

Keep records of which CA issues certificates for which domain: B. said that the idea, knowing which CA can issue certificates for which supplier, can have some additional value. However, the system will become less flexible. The importance and problems with updating the information was also mentioned.

One point of trust guarded by the counties government: In this case, the costs will be much higher in order to have a very high security. A normal hacker will not be able to attack anymore, however, big criminal organizations or another country with more money available will be able to, which will be an instant act of war. Every solution will have pros and cons.

PGP, ring of trust: This could for organizations be an solution where organizations trust each others self-signed certificates and by contract the responsibilities are recorded. Trust could for example be gained when companies do business together. The interviewee said that organizations will be less flexible to create a the ring of trust, due to trust matters.

DNSsec expansion to replace PKI: He said this solution is based on DNS and mentioned the problem of translating it to other domains, where DNS is not used. He said that the X.509 may be changed to serve for this purpose such that it will also work with this solution. However, the more you change, the higher the transition costs will become. If the structure of X.509 stays the same, the costs will be much less.

A.4. Dr. Benne de Weger - University Eindhoven

5 September 2012 at 16:00 - 17:00 in Eindhoven

The purpose of the interview was to discuss the possible solutions to minimalise the risks in PKI when a CA is compromised.

Benne de Weger is an assistant professor in the department of discrete mathematics in the field of cryptography at the Eindhoven University. He has great knowledge about PKI and has successfully issued an trusted certificate using collisions in the MD5 hash in 2008. He is now mostly a mentor for promovendi students, and before he was doing research on RSA.

PKI weaknesses

According to Benne, the weakest link in PKI are the organizational and procedural parts. Mistakes are made relatively easy in the organizational part. Technical and cryptographic mistakes are, like the MD5 collision they had found, very exceptional, he said.

MD5 hash collision attack

Benne and a student were able to create a trusted certificate in 2008 using a weakness in MD5, called the hash collision. Benne explained that they obtained a valid certificate trusted by a real CA. They did this by creating a bit string which had the same hash of a valid signature, which then could be used to sign another certificate. For this to work though, they needed a real CA to sign their hash. Benne said that if the technical procedures of that CA were not as predictable as they were, they would not have succeeded. The procedure by that CA was automated, without the interference of a human, with ordered serial numbers and timestamps. The exact timestamps was needed to create the certificate and if the procedure was not automated, they could not have guessed the right serial number and timestamps.

Attention points for designing improved PKI

It is important that the solution does not make it too difficult for the users, Benne said. That is where most good ideas fail.

Benne again mentioned that he is does not think of the current PKI badly.

A.4.1. Solutions

That a compromised trusted CA of all the CA's causes the complete network to be untrusted, could be said to be a design mistake, Benne said. He said that in the past years it has been made clear that PKI has some weaknesses, but does that make it worthless? He thinks not.

To decrease the risks, he said, it is important that CA's are open and transparent. But not only CA's, but also the audit companies need function better.

Certificates signed by 3 CA's

Benne said that the benefits, with 3 CA's signing a certificate, are that an entity without the rights to a certificate will have less chance to get one and that if one CA is compromised, not the complete network is not secure. Benne also mentioned the disadvantages for this solution. He said the solution will have increase the financial costs, also because getting a certificate will get more organizational. He said that the problem will be that services will develop that offer companies to apply the three certificates for them! Companies don't want to go to three different CA's to get their certificate signed. They will try to find another easier solution. This intermediary must be prevented if this solution is going to be implemented, Benne said.

He also mentioned the possibility to have a shared key by three CA's or group signatures by multiple CA's. This would enable us to implement the solution easier, because then the current X.509 can be used. Disadvantage is that these CA's need cooperate with each other.

Whitelisting next to blacklisting

Benne said that we already have this kind of solution and that it is called the Certificate Repository. He also said that even if this was more secure, the process is automated. If the hacker can control the process, he will also be able to add a certificate to the whitelisting. Or, the whitelisting must not be automated with a extra control. However, this control is already done during the registration, Benne said. In short, the solution does not really have an additional value.

Browser repositories with all the certificates and their CA's

Benne said that this is just another database that can get contaminated, but maybe it will work. This would divide the risk in compartments. Maybe an operational solution can be figured out, he said.

Perspective / Convergence

Benne mentioned the solution himself before I did, and said that PKI is broader than only the internet and this solution only works for the web. Benne also agreed that big criminal organizations could imitate this voting solution. Benne also mentioned that bandwidth could be a problem for this solution and that there should be some kind of business model to keep the notary servers online. People will not keep this up for too long.

DNSsec / DANE

This solution would also only work for the web, Benne said, but PKI is also used for email or code signing. But the trust in a software is not the same as the trust in a domain and he said that this may be a problem. He also agreed that ICANN will get too much of responsibility by being the only top root. Another problem with DNSsec as a solution is that it is too complex, which is also the reason why the implementation has taken so long. The biggest problem of DNSsec is that it facilitates Denial of Service, which could be a problem if DNSsec was also to replace PKI.

Benne also mentioned that DNS curve is another solution to solve the DNS spoofing attack, but is a new idea and not yet taken into account.

A.5. Pablo Valcárcel and Tanausú Cerdeña Hernández - Geosophic

10 September 2012 at 12:00 - 12:30 in Amsterdam

Pablo Valcárcel is the CEO and Tanausú Cerdeña Hernández the CTO of Geosophic, a company established in 2010. Geosophic is a gaming platform allowing mobile game developers to get behavior analytics from their players while offering the players new engagement triggers in the form of geolocated leaderboards.

Security at Geosophic

Tanausú said that because they offer an online service to game developers, security is important to them to assure the best user and developer experience. The security service, provided by Geosophic to game developers, is to secure the channel of communication over the 3G network. To accomplish this they make use of certificates. Geosophic also saves data on their servers, which need to be in compliance with the privacy law for saving private data.

Does the usage of certificates slow the communication?

Tanausú said that it is a small overhead; 5% to 10% of the total speed. Although Geosophic does not transmit personal data, using certificates is neccessary, because you have to be sure the data is encrypted. If the data is not encrypted, you may get big problems, he said.

Cheating

Precautions against cheating have been considered, but currently not yet implemented. They do have behaviour detection against cheating where for example, when a user tries too many times to log in with different passwords, this user can be blocked.

However, because the games are not developed by Geosophic, they have little control over cheating in those games.

Leaderboard

Tanausú mentioned that they provide a feature called the leaderboard to the game developers, where the player's scores are saved and delivered to other players. A possible way of cheating would be if someone sends in fake scores, for instance to become the top in the leaderboard. To prevent this from happening they try to detect this and inform the developer. This can for example be done using the unique ID of the player and the number of times the player sends in highscores. If this is too often, an 'alarm' is triggered meaning it could be someone who is trying to cheat.

Tanausú agreed with me that if someone sends a plausible fake highscore to the server only once, then it would not be detected, because they do not want to prevent scores automatically, due to possible mistakes in the algorithms. If automaticly done, the possibility exists that a real score is removed from the leaderboard. Pablo said that it is better to have somebody that cheats occasionally than to limit a player with a proper score.

Leaderboard and certificates

The problem here is that every player then will need a certificate, which is not very easy. Tanausú said he thinks this is not feasible for now. This could be done if it were something more critical, like finance information, but for highscores, this is sufficient.

Tanausú agreed with me that even then highscores could be signed by the cheater, but said that this would be more difficult to accomplish, because the cheater then needs to obtain your private key. Tanausú said: "We will never be a 100% safe, but try to make it as safe as we can"

Problems using certificates

Tanausú said they did not have had any problems using certifcates. The certificates are not used to sign contents, but used to secure the communication channel, where the certificate is put on the serverside, enabling HTTPS.

Mobile game security

Tanausú added that for most mobile games security is not required for the users. It would be an issue if a lot of money was spent by users, such as in big MMORPG games. Gambling games were also mentioned by the team, saying that security must be very important to gambling companies due to the high amount of money that is spent by the users. They also mentioned that gambling companies need to know who the player really is, and specifically the player's age.

Spain

In Spain a lot of people have their own ID-card, containing a private key, such that the people can for example do their taxes online. The technical solution is not perfect, but the idea behind it is very good, Tanausú said.

A.6. Co-founder - Game company

13 September 2012 at 11:30 - 12:00

The interviewee, C., is the Co-founder of the game company, a Dutch game development studio of casually connected games for the latest generation of console and mobile platforms.

Security at the company

C. said that in their games, in general, the security is done by the SDK of the platform they use. In XBOX for example, security is managed by Microsoft and for PlayStation he expects the same thing. He explained that after their data is sent to the SDK, the data sent from the SDK to the platform's servers is encrypted.

To the question if the company has used or implemented their own security, he said that once it was required to use a tool from a third party. He said that if they were going to use their own security, it will mostly be a out of the box tool. This is not their business core and will probably take too much time and money for them. He said that in most cases it is not even necessary to implement security or know about security, because in most cases this is delivered by the supplier and they are doing

a good job at this. For example, the achievements tracked by Microsoft is not yet compromised.

Source code protection

To the question if they had any measurements for source code security, he answered that this also is done by the platform's SDK, and that they, in most cases, only use the tools in those libraries.

The company does not use DRM, and C. mentioned that the Ubisoft online DRM which is used by some games has the disadvantage that games can only be played online, which is not really desired.

Facebook - Privacy

C. explained that the projects done by the company for Facebook are not very big to have security against cheaters. They do try to make the code as secure as possible and difficult to understand for hackers. He said that they do use certificates for HTTPS to encrypt the communications. He also said that some of the data of the users, such as passwords, is encrypted before it is saved in the database.

Scoreboard problem

After I mentioned the problem with the scoreboard, where a hacker could simply send his altered score to the server, Maurice said that this was an interesting problem. With another game of theirs they tried to make this more difficult by generating ID's which then was used to encrypt the score. This was an extra security check but not a waterproof security measurement, because the end user could still use this ID to encrypt the altered score.

A.7. Game hacker

2 October 2012 at 12:00 - 12:30

D. is a bachelor student and has been hacking games since he was a child. Meanwhile he has 7 years of experience with hacking games. It should be noted that by hacking, he means exploring and finding out how the software works and not abusing the knowledge.

The problem with game security, he said, is that the game developer has to give all the rights to the player's computer: the code, the certificates... everything. In this case, if the player is a hacker and this hacker has enough time and intelligence to figure out the game, it is a lost battle for the game developer. D. said that until now, he has not found a perfect solution for the game developers. Even if the communication is secured using certificates, the computer of the hacker must still be able to decrypt them. Then of course, the hacker, depending on his skills, will also be able to find the keys, decrypt, and modify the messages.

When have you reached your goal?

He never reaches his goal, he said, but in time loses his motivation when he knows how the important parts of the game work; Such as the encryption, the allowance to do something illegal like walking faster, or when he has documentated the basics of the game. For League of Legends, an multiplayer online battle game, for example, he lost his motivation after he was able to set up a private server where players could walk, talk, and some other things. The rest is not interesting to him, because that is just further exploiting the information he already has. He now knows that LoL uses blowfish; receives the key from the server, which is sent to the Game Client through the Air Client, which is started before the actual Game Client, using command line. Knowing this allows you to get the key and decrypt packets.

Is getting the highscore not a goal?

D. said his goal is not to get the highscore, but to figure out how the game works, like which encryption and algorithms are used, what the key is, how he could reverse it and use it. He does not need to become the best player. He said he had never used his skills to get a highscore or a prize.

Achievements in Steam (Valve):

D. said he had never tried to do that, because that is not his goal. However, he is sure it wouldn't be too hard. Steam gives achievements after some particular actions have happened. There are two ways to get this done, he said: One option is to modify the packets yourself, while the other is to run the command in the game responsible for actions such as jumping. For headshots you could do the same, find the function that registers the headshots, and run that. No need to actually make the headshots.

Hacking a platform game:

Microsoft has chosen the extension .exe, Linux uses ELF, the PlayStation has something similar to Linux, the Xbox uses the same as windows, and for Nintendo he wasn't sure, he said. He did convert his PlayStation 2, but has not tried to hack a platform game, because that costs too much of an effort. For PlayStation for example, you first need to decrypt and modify the CD. After modification, the PlayStation can't read it anymore, so extra steps are needed. After many steps, you can actually start looking into the game, which is much more effort than with PC games, where you have full rights if it is your own computer.

A.7.1. Possible security solutions and their drawbacks

Restricting the play of the game to online only:

Blizzard has designed a clever model with Diablo 3, where players can only play the game if they play it online. If not played online, the game will not save the data, give the player achievements etc.

A possible hack to this security measurement, D. said, is if the hacker mimics the server. The private server could allow players to connect and play the game, without the interference of the original game server. Drawback to this attack is that the achievements and scores are not achieved in the official server. It does allow the game to be played without the monthly fee such as in the World of Warcraft (WoW).

D. himself, pays the fee to play WoW. He said that playing on the official servers is more fun, due to the amount of players and the events that WoW creates during holidays. Also, he thinks Ubisoft is making a smart move by changing their model and making their games free-to-play, where players pay for extra's inside the game.

Keep some parts of the software on the server:

In windows, this could be done by sending the information in the stack to the server, to then let the server perform some operations on the information and send the results back, either encrypted or not.

The problem here, is that when the results are received, the software still needs to unpack and/or decrypt the results and do something with it. This is always the weakest part, because, if the computer can decrypt and read the results, so can the hacker, or will able in time.

Calculate the game on the server:

D. he discovered that the security of LoL was by far the best security he had seen. The server calculates everything alongside the game client. In LoL, the player's computer does not send the location, but the direction the player is going to. If the calculated positions of the server and the gamer's computer don't overlap, the server will determine the character's position.

D. mentioned one problem: LoL is an online game and which always has a latency problem. The hacker can abuse this and walk for instance faster with a factor of 0,01. The server will not notice this and in a game such as LoL, such small differences matter.

Cloudgaming:

D. mentioned this is when the game, the executable, is placed on the server and only keystrokes of the player is sent to the server. The server, in turn, only sends the screen to the player. This is possible with today's internet, he said, and would be the only real solution for companies, because then hackers can't modify or learn about the source code.

Secure the connection with certificates:

Encrypting the communication is good and makes it more difficult to read or modify the data, but the hacker could still use reverse engineering and read the keys of the certificate.

Make the game code unreadable for the hacker:

This method is called obfuscation, which makes the assembly code unreadable by spreading the code everywhere. This makes it very difficult to find the flow of the data. It is still possible to read the code, but much more time consuming.

A drawback of this method is that the software will not be efficient, because the code is not written in a optimal way.

Checking the data after it is sent to the server:

This kind of security checks if the player has not tampered witht the data. If so, the player can be banned from the game when detected. In WoW, they have a legion of game masters who check the players. He said he tried to use a fish bot himself (which actually fishes fish for the player), but WoW detected this immediately. If something is not as the normal player data, they can detect this while a human double checks.

A.7.2. How do you perform reverse engineering

D. said there are two possible usages for reverse engineering: the programs and the packages sent over network. For online games, he could use programs such as Wireshark, but he has created his own, opensource, advanced program to reverse engineer the network packages. His program, controlled through a front-end, creates a back-end for each application. The back-end can for example install a jump in the code to let the game jump to his program after an amount of steps. This is one of the many attacks that can be used.

Professional tools he could use to reverse engineer programs are: IDA Professional, but this software costs a lot. Another software is OllyDbg, which is opensource, but is developed slowly.

A.7.3. Certificates

After thinking about where certificates could be used for, inside games, D. mentioned that he had experiences with a company, not a game company though. They sell PHP code/websites and allow the buyer to download the source code, which is encrypted and will only work after checking the validity of the bought license. To bypass this, D. redirected the part that checked the license to his own servers, which always replied positively to the requests. A solution would be to have PKI used for this, where the server would need to identify itself as the real license server which would disable the hack.

A.8. Game developer - Game company

3 October 2012 at 12:00 - 12:30

E. has been working as a game developer at the company for 1,5 year now and is also responsible for the security parts of their game. He also contributes a lot to the game engine and game play features.

A.8.1. Security Issues

E. explained that there are 4 important security issues that need to be dealt with. The most important issue is to prevent cheaters in their game, their online multiplayer game. The second problem is the interception of network traffic between the server and client by a cheat program, which can be used to alter the packages and ultimately to cheat in the game. The third problem is the modification of the files, allowing the player to, for instance, do more damage or have more health. A similar problem is the modification of saved games, which saves the process of the character. However, most of this is saved online, making it harder for hackers to modify.

A.8.1.1. Game memory modification

There are cheat programs available that can fairly easy modify the memory such as the level of the player's health. These programs check while the game is running, where the memory changes when, for instance, the health is changed and modify these numbers accordingly. There is a program called "Cheat Engine", which does this. The game company tries to make the location of the numbers more difficult to be found in the memory. There are multiple tricks to accomplish this. Every time the game is updated, the previous cheats won't work, but after only a few days there are cheats available again. The updates are distributed and released by Steam, a platform for distributing games. The new files and executable are uploaded which then is given to the players through Steam.

A.8.1.2. Network interception

Encryption is used to disallow the modification of packets sent over the network. For this, they don't use certificates issued by CA's, but have their own asymmetric and symmetric cryptography. Every round in their game is played by 6 players. For each client, every time the game starts, the game generates a public and a private key, which is distributed to the all the other players of that round. The public keys is then used to exchange a symmetric key for each pair of players. Thus, each player receives 5 symmetric keys. The 6 players are connected to each other using Steam's match making system. In this system, one of the players acts as the server and if the player leaves, the server is handed over to another player. E. said proudly that their key method was not hacked yet! He was, however, convinced that a good hacker will be able to find out how and where the keys are stored.

A.8.1.3. File modification

Another problem is the modification of the game files. E. said that as long as the software runs on the player's computer, the hacker will be able to eventually figure out the game. The only possible solution is to make it harder for them. Obfuscation, which is making the code more difficult for the hacker to read, is one way. Most of the time, however, this method will have consequences for the performance of the game. There are different types of obfuscation. Some will translate the code to byte code and run an interpreter over it, which makes the code very difficult to read. E. said, that obfuscation is also used by virus developers to hide their virus, resulting that anti-virus programs recognize such codes as a virus. E. had tried this out and indeed found that the game was recognized as a virus, making this an undesirable solution.

Another method to make it harder for the hacker is to not use standard encryption algorithms, but to add a modified version to a strong algorithm. The key can eventually be founded inside the code using reverse engineering. Using AES in combination with something self-designed for example, makes it harder for the hacker to figure out how to decrypt the messages using the key.

By using reverse engineering the hacker is not able to get the original code, but the hacker could read the machine code. A measurement taken by Steam is that a hash is added to the game and the game won't start without this hash. So if the game is modified, this hash still is needed to allow the start up. After the game has started, the game could be altered though. Measurements against this, like checking every second, will make the game less efficient and therefore not desirable.

A.8.2. Banning cheaters

Their game does have a system to ban players. Cheaters are not recognized automatically though, but are reported by other players. They hope the bans will scare off players to use cheats.

The game company has thought of using an automatic system that monitoring players and detecting cheats, however, this would have cost too much time and effort, making it not worth implementing for a small game studio. They do keep statistics of the games, but just a small amount of data which can't be used for cheat detection.

Another Steam cheat prevention mechanism is implemented in Steam and checks if a player is running a cheat program during the game. If so, the player's steam account will eventually be banned. Concerning privacy of users, this is mentioned in Steam's Terms and Conditions, the data is anonymous, and it is not uploaded to Steam's servers. Steam only checks if the player is running such program.

A.8.3. Platforms

Cheating in the platforms is much more difficult and almost not done. The only cheating problems is on the PC and has cost already a lot of time. However, there are also more players on PC playing their game. Note that players from different platforms can not play with each other.

A funny property of PlayStation and Xbox to mention, is that the in-game voice chatting is not allowed to be encrypted in the U.S., because then it could be used by terrorists to communicate. Fortunately, this is done automatically by the libraries of the platforms.

A.8.4. Which data would be put in the cloud, if this was possible?

All the data that has something to do with the game play needs to be put in the cloud, if this was to be implemented. However, E. answered that implementing this would not pay off for them, because the players are all over the world, meaning they would need multiple cloud servers geographically spread. Otherwise it would cause too much lag for the players. The peer-to-peer network is the only possibility right now.

Bibliography

[ABD+00] Carlisle Adams, Mike Burmester, Yvo Desmedt, Mike Reiter, and Philip Zimmermann. Which pki (public key infrastructure) is the right one? (panel session). In *Proceedings of the 7th ACM conference on Computer and communications security*, CCS '00, pages 98–101, New York, NY, USA, 2000. ACM. URL: http://doi.acm.org/10.1145/352600.352615, doi:10.1145/352600.352615.

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, January 2004. URL: http://dx.doi.org/10.1109/TDSC.2004.2, doi:10.1109/TDSC.2004.2.

[Ark12] Brad Arkin. Inappropriate Use of Adobe Code Signing Certificate. blogs.adobe.com, September 2012. URL: http://blogs.adobe.com/asset/2012/09/inappropriate-use-of-adobe-code-signing-certificate.html.

[Ass12] Fabio Assolini. Brazilian Trojan bankers now digitally signed. securelist, September 2012. URL: http://www.securelist.com/en/blog/208193825/brazilian_trojan_bankers_now_digitally_signed.

[BBB+07] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. NIST Special Publication 800-57. *NIST Special Publication*, 800(57):1–142, 2007.

[Bel95] Steven M. Bellovin. Using the Domain Name System for system breakins. In *Proceedings of the 5th conference on USENIX UNIX Security Symposium - Volume 5*, SSYM'95, pages 18–18, Berkeley, CA, USA, 1995. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1267591.1267609.

[Ben01] M. Benantar. The internet public key infrastructure. *IBM Syst. J.*, 40(3):648–665, March 2001. URL: http://dx.doi.org/10.1147/sj.403.0648, doi:10.1147/sj.403.0648.

[Ben11] Simon Bennetts. Colour map of CAs. EFF.org, September 2011. URL: https://www.owasp.org/index.php/File:Colour_map_of_CAs.pdf.

[Bra00] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[CDFT98] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format. Technical report, RFC 2440, November, 1998.

[Cen12] Nationaal Cyber Security Centrum. Veilig beheer van digitale certificaten. www.ncsc.nl, September 2012. 27 sept.

[Cil12] Chelate Cilraaz. Authenticators and security. mmo-champion.com, May 2012. 2012-05-31. URL: http://www.mmo-champion.com/threads/1139962-Authenticators-and-security.

[CNS+08] D. Cooper, NIST, S. Santesson, Microsoft, S. Farrell, Trinity College Dublin, S. Boeyen, Entrust, R. Housley, Vigil Security, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile RFC 5280, May 2008. URL: http://datatracker.ietf.org/doc/rfc5280/?include_text=1.

[com] Comcast DNSSEC validation. dns.comcast.net.

[Com11a] Comodo. Comodo fraud incident 2011-03-23. www.comodo.com, March 2011. URL: http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.

[Com11b] ComodoHacker. Another status update message. Pastebin.com, September 2011. URL: http://pastebin.com/85WV10EL.

[Com11c] ComodoHacker. Striking Back... Pastebin.com, September 2011. september 5. URL: http://pastebin.com/1AxH30em.

[Con] Running a Notary. Github. URL: https://github.com/moxie0/Convergence/wiki/Running-a-Notary.

[Con11] Lucian Constantin. EFF proposes new method to strengthen Public Key Infrastructure. PCWorld, November 2011. URL: http://www.pcworld.idg.com.au/article/408227/eff_proposes_new_method_strengthen_public_key_infrastructure/.

[CTwa] Certificate Transparency. sites.google.com. URL: https://sites.google.com/site/certificatetransparency/.

[CTwb] Certificate Transparency - How the logs work. www.certificate-transparency.org. URL: http://www.certificate-transparency.org/home/how-the-logs-work.

[CXTL02] Wentong Cai, Percival Xavier, Stephen J. Turner, and Bu-Sung Lee. A scalable architecture for supporting interactive games on the internet. In *Proceedings of the sixteenth workshop on Parallel and distributed simulation*, PADS '02, pages 60–67, Washington, DC, USA, 2002. IEEE Computer Society.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 1976. URL: http://dx.doi.org/10.1109/TIT.1976.1055638, doi:10.1109/TIT.1976.1055638.

[Dig12] Het DigiNotarincident. De onderzoeksraad voor veiligheid, June 2012. Projectnummer S20110V0902-03.

[DK01] Hans Delfs and Helmut Knebl. *Introduction to cryptography: principles and applications*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[DKL+04] Margaret DeLap, Björn Knutsson, Honghui Lu, Oleg Sokolsky, Usa Sammapun, Insup Lee, and Christos Tsarouchis. Is runtime verification applicable to cheat detection? In *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, NetGames '04, pages 134–138, New York, NY, USA, 2004. ACM. URL: http://doi.acm.org/10.1145/1016540.1016553, doi:10.1145/1016540.1016553.

[DNS12] The DANE Protocol: What It Is And Why It Matters (including DNSSEC, SSL/TLS). Youtube.com, May 2012. URL: https://www.youtube.com/watch?feature=player_embedded&v=emDxUQl1NvA.

[EB10] Peter Eckersley and Jesse Burns. An Observatory for the SSLiverse. EFF.org, 2010. URL: https://www.eff.org/files/DefconSSLiverse.pdf.

[ECH08] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1065–1074, New York, NY, USA, 2008. ACM. URL: http://doi.acm.org/10.1145/1357054.1357219, doi:10.1145/1357054.1357219.

[Eck11a] Peter Eckersley. 28c3: Sovereign Keys. Youtube.com, December 2011. URL: http://www.youtube.com/watch?v=18pFTo3zVxk.

[Eck11b] Peter Eckersley. Consensus on timelines. git.eff.org, November 2011. URL: https://git.eff.org/?p=sovereign-keys.git;a=blob;f=issues/consensus-on-timelines.txt;h=bf39abb2897528ae04af5eb730aa52ea1c2949b1;hb=HEAD.

[Eck11c] Peter Eckersley. How secure is HTTPS today? How often is it attacked?, October 2011. URL: https://www.eff.org/deeplinks/2011/10/how-secure-https-today.

[Eck11d] Peter Eckersley. Sovereign Key Cryptography for Internet Domains. git.eff.org, November 2011. URL: https://git.eff.org/?p=sovereign-keys.git;a=blob_plain;f=sovereign-key-design.txt;h=5b90d1174444ffc273d5dbd22d11b55cdb9d884b;hb=HEAD.

[Eck11e] Peter Eckersley. Sovereign Keys - A proposal for fixing attacks on CAs and DNSSEC. 28th Chaos Communication Congress, December 2011. URL: http://events.ccc.de/congress/2011/Fahrplan/events/4798.en.html.

[Eck11f] Peter Eckersley. Sovereign Keys: A Proposal to make HTTPS and Email More Secure. eff.org, November 2011. URL: https://www.eff.org/deeplinks/2011/11/sovereign-keys-proposal-make-https-and-email-more-secure.

[Eck11g] Peter Eckersley. Transitional-considerations. git.eff.org, November 2011. URL: https://git.eff.org/?p=sovereign-keys.git;a=blob;f=issues/transitional-considerations.txt;h=fa3b1591820baf1f2f62740f1f0e8b7998c29174;hb=HEAD.

[EP11] C. Evans and C. Palmer. Public Key Pinning extension for HTTP. tools.ietf.org, November 2011. URL: http://tools.ietf.org/html/draft-ietf-websec-key-pinning-01.

[FB00] Warwick Ford and Michael S. Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2000.

[Fis] Fishbot. www.fishbot.net. URL: http://www.youtube.com/watch?feature=player_embedded&v=huT3UIEPOjA.

[FW98] Jalal Feghhi and Peter Williams. *Digital Certificates: Applied Internet Security (with CD-ROM)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[Gam11] Andy Gambles. Fraudulent Comodo SSL Certificates Issued. ServerTastic Blog, June 2011. URL: http://blog.servertastic.com/fraudulent-comodo-ssl-certificates-issued/.

[Gra11] David Graham. The Comodo hacker releases his manifesto. Errata Security, March 2011. 27 March 2011. URL: http://erratasec.blogspot.com.es/2011/03/comodo-hacker-releases-his-manifesto.html.

[Gut02] Peter Gutman. PKI: It's Not Dead, Just Resting. *Computer*, 35(8):41–49, August 2002. URL: http://dx.doi.org/10.1109/MC.2002.1023787, doi:10.1109/MC.2002.1023787.

[GZLM04] Chris GauthierDickey, Daniel Zappala, Virginia Lo, and James Marr. Low latency and cheat-proof event ordering for peer-to-peer games. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '04, pages 134–139, New York, NY, USA, 2004. ACM. URL: http://doi.acm.org/10.1145/1005847.1005877, doi:10.1145/1005847.1005877.

[HCS+12] P. Hoffman, VPN Consortium, J. Schlyter, , and Kirei AB. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA RFC 6698. IETF.org, August 2012.

[HPJ12] J. Hodges, PayPal, and C. Jackson. HTTP Strict Transport Security (HSTS). IETF, November 2012. URL: https://www.ietf.org/rfc/rfc6797.txt.

[Kar94] C.M. Karat. *A business case approach to usability cost justification.* Academic Press, New York, 1994.

[Koe12] Chris Koenis. SSL-aanbieders: piratensites zijn niet te voorkomen. Webwereld.nl, July 2012. 19 juli. URL: http://webwereld.nl/nieuws/111219/ssl-aanbieders--piratensites-zijn-niet-te-voorkomen.html.

[Koh78] L.M. Kohnfelder. *Towards a Practical Public-key Cryptosystem.* Massachusetts Institute of Technology, 1978. URL: http://books.google.nl/books?id=eZcttwAACAAJ.

[Lan11a] A. Langley. Certificate Transparency. ImperialViolet, November 2011. URL: http://www.imperialviolet.org/2011/11/29/certtransparency.html.

[Lan11b] A. Langley. Classifying solutions to the certificate problem. ImperialViolet, October 2011. URL: http://www.imperialviolet.org/2011/10/08/certaxis.html.

[Lan11c] Adam Langley. Certificate Transparency. Google+, November 2011. URL: https://plus.google.com/u/1/118082204714636759510/posts/NZ8NrLXx4Uj.

[Lan11d] Adam Langley. DNSSEC authenticated HTTPS in Chrome. Imperial Violet, June 2011. URL: http://www.imperialviolet.org/2011/06/16/dnssecchrome.html.

[Lan11e] Adam Langley. Public Key Pinning. Imperial Violet, May 2011. URL: http://www.imperialviolet.org/2011/05/04/pinning.html.

[Lan11f] Adam Langley. Why not Convergence? ImperialViolet - Adam Langley's Weblog, September 2011. URL: http://www.imperialviolet.org/2011/09/07/convergence.html.

[Lan12] Edwin Lankamp. DigiNotat - What Went Wrong? www.css-security.com, July 2012. URL: http://www.css-security.com/blog/diginotar-what-went-wrong/.

[Lau12] Ben Laurie. Revocation Transparency and Sovereign Keys. Links.org, September 2012. URL: http://www.links.org/?p=1272.

[LDMW04] Kang Li, Shanshan Ding, Doug McCreary, and Steve Webb. Analysis of state exposure control to prevent cheating in online games. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '04, pages 140–145, New York, NY, USA, 2004. ACM. URL: http://doi.acm.org/10.1145/1005847.1005878, doi:10.1145/1005847.1005878.

[Lee05] J. Lee. Wage Slaves. 1UP.com, May 2005.

[Ley12] John Leyden. One in four don't clean their stinky old browsers - especially Firefoxers. The Register, November 2012. URL: http://www.theregister.co.uk/2012/11/12/outdated_browser_software_kaspersky/.

[LK12] Ben Laurie and Emilia Kasper. Certificate Transparency v2.1a. Links.org, September 2012. URL: http://www.links.org/files/CertificateTransparencyVersion2.1a.pdf.

[LL11] Ben Laurie and Adam Langley. Certificate Authority Transparency and Auditability. www.links.org/, November 2011. URL: http://www.links.org/files/CertificateAuthorityTransparencyandAuditability.pdf.

[LPBC07] Peter Laurens, Richard F. Paige, Phillip J. Brooke, and Howard Chivers. A Novel Approach to the Detection of Cheating in Multiplayer Online Games. In *Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems*, ICECCS '07, pages 97–106, Washington, DC, USA, 2007. IEEE Computer Society. URL: http://dx.doi.org/10.1109/ICECCS.2007.11, doi:10.1109/ICECCS.2007.11.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X. 509 Internet public key infrastructure online certificate status protocol-OCSP. Technical report, RFC 2560, 1999.

[Mar11a] M. Marlinspike. Blackhat USA in Las Vegas, 2011. URL: https://www.blackhat.com/html/bh-us-11/bh-us-11-speaker_bios.html.

[Mar11b] Moxie Marlinspike. Video - SSL And The Future Of Authenticity. Defcon Las Vegas, August 2011. URL: https://media.defcon.org/dc-19/video/DEF%20CON%2019%20Hacking%20Conference%20Presentation%20By%20-%20Moxie%20Marlinspike%20-%20SSL%20And%20The%20Future%20Of%20Authenticity%20-%20Video.m4v.

[MGM06] Christian Mönch, Gisle Grimen, and Roger Midtstraum. Protecting online games against cheating. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, NetGames '06, New York, NY, USA, 2006. ACM. URL: http://doi.acm.org/10.1145/1230040.1230087, doi:10.1145/1230040.1230087.

[Mic02] S. Micali. Scalable certificate validation and simplified pki management. In *1st Annual PKI Research Workshop*, page 15, 2002.

[Mor12a] Mike Morhaime. Important security update. us.Blizzard.com, August 2012. spelers gingen dood - directeur Mike heeft gereageerd. URL: http://us.blizzard.com/en-us/securityupdate.html.

[Mor12b] Bruce Morton. Certificate Transparency Birds of a Feather. SSL.Entrust, November 2012. URL: http://ssl.entrust.net/blog/?p=1605.

[Nes12] Jonathan Ness. Microsoft certification authority signing certificates added to the Untrusted Certificate Store. blogs.technet.com, June 2012. 3 June 2012. URL: http://blogs.technet.com/b/srd/archive/2012/06/03/microsoft-certification-authority-signing-certificates-added-to-the-untrusted-certificate-store.aspx.

[NN00] Jakob Nielsen and Donald A. Norman. Web-Site Usability: Usability On The Web Isn't A Luxury, January 2000. URL: http://www.informationweek.com/773/web.htm.

[NPVS07] Christoph Neumann, Nicolas Prigent, Matteo Varvello, and Kyoungwon Suh. Challenges in peer-to-peer gaming. *SIGCOMM Comput. Commun. Rev.*, 37(1):79–82, January 2007. URL: http://doi.acm.org/10.1145/1198255.1198269, doi:10.1145/1198255.1198269.

[NST09] M. Narasimha, J. Solis, and G. Tsudik. Privacy-preserving revocation checking. *International Journal of Information Security*, 8(1):61–75, 2009.

[OW11] Eric Ouellet and Vic Wheatman. X.509 Certificate Management: Avoiding Downtime and Brand Damage. Gartner, November 2011.

[Pag11] Paganinip. SSL replacement? Convergence for replacing CA... maybe, November 2011. URL: http://securityaffairs.co/wordpress/151/digital-id/ssl-replacement-convergence-for-replacing-ca-maybe.html.

[Pal11] Cevans Palmer. Dynamic Public Key Pinning. tools.ietf.org, November 2011. URL: https://tools.ietf.org/agenda/82/slides/websec-1.pdf.

[PE11] Jesse Burns Peter Eckersley. The (Decentralized) SSL Observatory. USENIX Security 2011, 2011. URL: http://static.usenix.org/events/sec11/tech/slides/eckersley.pdf.

[PHBL12] R. Stradling P. Hallam-Baker, Comodo Group Inc. and Comodo CA Ltd. DNS Certification Authority Authorization (CAA) Resource Record - draft-ietf-pkix-caa-15. IETF.org, October 2012. URL: http://tools.ietf.org/html/draft-ietf-pkix-caa-15.

[Pin08] et al. Pinkas. CMS Advanced Electronic Signatures (RFC 5126). IETF, February 2008. URL: http://www.ietf.org/rfc/rfc5126.txt.

[Pin12] D. Pinkas. X. 509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP. 2012.

[Pri00] M. Pritchard. How to hurt the hackers: The scoop on internet cheating and how you can combat it. *Gamasutra, July*, 24, 2000.

[Pri11] J. R. Prins. Interim Report: Operation Black Tulip. Public, Fox-IT, Olof Palmestraat 6, Delft, The Netherlands, September 2011.

[Ric12] Robert Richardson. Will TurkTrust incident raise certificate use to Chrome standard? SearchSecurity.techtarget.com, January 2012. URL: http://searchsecurity.techtarget.com/news/2240175732/Will-TurkTrust-incident-raise-certificate-use-to-Chrome-standard.

[Riv90] Ronald L. Rivest. Handbook of theoretical computer science (vol. A). In Jan van Leeuwen, editor, *Handbook of theoretical computer science*, chapter Cryptography, pages 617–755. MIT Press, Cambridge, MA, USA, 1990. URL: http://dl.acm.org/citation.cfm?id=114872.114885.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. URL: http://doi.acm.org/10.1145/359340.359342, doi:10.1145/359340.359342.

[Sch11] Bruce Schneier. The EFF's Sovereign Key Proposal. www.schneier.com, December 2011. URL: http://www.schneier.com/blog/archives/2011/12/the_effs_sovere.html.

[SEA+09] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: an empirical study of SSL warning effectiveness. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 399–416, Berkeley, CA, USA, 2009. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1855768.1855793.

[SID12] Dnssec. SIDN.nl, 2012. URL: https://www.sidn.nl/over-nl/dnssec/.

[SKS] Project - Sovereign-Keys.git. git.eff.org. URL: https://git.eff.org/?p=sovereign-keys.git;a=tree;hb=HEAD.

[SKs11] Sybil attack by mirrors, 2011. URL: https://git.eff.org/?p=sovereign-keys.git;a=blob;f=issues/sybil-attack-by-mirrors.txt;h=d9f8baf6affa2552567a4d091752cec44e219071;hb=HEAD.

[Sta12] Forum Standaardisatie. DNSSEC | Open Standaarden, June 2012. URL: https://lijsten.forumstandaardisatie.nl/open-standaard/dnssec.

[Sto] James Stone. How to Change the Certificate Authority. eHow Contributor. URL: http://www.ehow.com/how_7692417_change-certificate-authority.html.

[Sys04] Cisco Systems. Public Key Infrastructure Certificate Revocation List versus Online Certificate Status Protocol. Technical report, Cisco Systems, 2004.

[Tel02] G. Tel. *Beveiliging van de digitale maatschappij*. Addison Wesley, 2002.

[Tew12] Erik Tews. Sovereign Keys - A proposal for fixing attacks on CAs and DNSSEC. cryptanalysis.eu, January 2012. Erik is a dr. in Security Engineering Group TU Darmstadt. URL: http://cryptanalysis.eu/blog/2012/01/18/sovereign-keys-a-proposal-for-fixing-attacks-on-cas-and-dnssec/.

[Vas12] Vasco.com. Leading online game operators trust VASCO's solutions and expertise to secure user's accounts. 2012. URL: http://www.vasco.com/Images/e-Gaming.pdf.

[vM12] Olaf van Miltenburg. Aantal met DNSSEC ondertekende .nl-domeinen groeit tot boven 600.000. Tweakers.net, August 2012. URL: http://tweakers.net/nieuws/83827/aantal-met-dnssec-ondertekende-nl-domeinen-groeit-tot-boven-600000.html.

[WAP08] Dan Wendlandt, David G. Andersen, and Adrian Perrig. Perspectives: improving SSH-style host authentication with multi-path probing. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 321–334, Berkeley, CA, USA, 2008. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1404014.1404041.

[Wei01] Joel Weise. Public Key Infrastructure Overview. http://www.sun.com/blueprints, August 2001.

[Wen08] Dan Wendlandt. Perspectives: Can Host Authentication be Secure AND Cheap? The North American Network Operators' Group, October 2008. URL: http://www.nanog.org/meetings/nanog44/presentations/Tuesday/Wendlandt_Perspectives_usenix_N44.pdf.

[Wie] Jered Wierzbicki. Sovereign Key Draft Specification. Git.eff.org. URL: https://git.eff.org/?p=sovereign-keys.git;a=blob;f=spec.txt;h=b07dc1a1506a2bdd1b169c8a4ee3b994c562dede;hb=HEAD.

[WS07] S.D. Webb and S. Soh. Cheating in networked computer games: a review. In *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, pages 105–112. ACM, 2007.

[Yan03] Jeff Yan. Security Design in Online Games. In *Proceedings of the 19th Annual Computer Security Applications Conference*, ACSAC '03, pages 286–, Washington, DC, USA, 2003. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=956415.956453.

[YC02] J.J. Yan and H.J. Choi. Security issues in online games. *Electronic Library, The*, 20(2):125–133, 2002.

[YLLY06] S. Yeung, J.C.S. Lui, J. Liu, and J. Yan. Detecting cheaters for multiplayer games: theory, design and implementation. In *Proc IEEE CCNC*, volume 6, pages 1178–1182, 2006.

[YR05] Jeff Yan and Brian Randell. A systematic classification of cheating in online games. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, NetGames '05, pages 1–9, New York, NY, USA, 2005. ACM. URL: http://doi.acm.org/10.1145/1103599.1103606, doi:10.1145/1103599.1103606.